Overview and Structure

22.10.2014 Organisation
22.10.3014 Introduction (Def.: Machine learning, Supervised/Unsupervised, Examples)
29.10.2014 Machine Learning Basics (Toolchain, Features, Metrics, Rule-based)
05.11.2014 A simple Supervised learning algorithm
12.11.2014 Excursion: Avoiding local optima with random search
19.11.2014 –
26.11.2014 Bayesian learner
03.12.2014 –
10.12.2014 Decision tree learner
17.12.2014 k-nearest neighbour
07.01.2015 Support Vector Machines
14.01.2015 Artificial Neural Networks and Self Organizing Maps
21.01.2015 Hidden Markov models and Conditional random fields
28.01.2015 High dimensional data, Unsupervised learning
04.02.2015 Anomaly detection, Online learning, Recommender systems

Machine Learning and Pervasive Computing
Outline

Anomaly detection

Recommender systems

Stochastic Classification

Online learning
Anomaly detection

Problem statement

Anomaly detection is the task to identify anomalous behaviour with respect to behaviour trained on a data set of prior samples.
Anomaly detection

Problem statement

Anomaly detection is the task to identify anomalous behaviour with respect to behaviour trained on a data set of prior samples

Examples: Fraud detection Identify users with unusual behaviour (e.g. non-human profiles in a web-application)
Anomaly detection

Problem statement

Anomaly detection is the task to identify anomalous behaviour with respect to behaviour trained on a data set of prior samples

Examples: 
- **Fraud detection**: Identify users with unusual behaviour (e.g. non-human profiles in a web-application)
- **Manufacturing**: Identify faulty engines (e.g. too much vibration, too loud, ...)


Anomaly detection

Problem statement

Anomaly detection is the task to identify anomalous behaviour with respect to behaviour trained on a data set of prior samples.

Examples:

**Fraud detection**  Identify users with unusual behaviour (e.g. non-human profiles in a web-application)

**Manufacturing**  Identify faulty engines (e.g. too much vibration, too loud, ...)

**Computers in a datacenter**  Identify machines that do not work properly (e.g. processing load over network traffic)
Anomaly detection

Problem statement

Anomaly detection is the task to identify anomalous behaviour with respect to behaviour trained on a data set of prior samples.

**Examples:**

**Fraud detection** Identify users with unusual behaviour (e.g. non-human profiles in a web-application)

**Manufacturing** Identify faulty engines (e.g. too much vibration, too loud, ...)

**Computers in a datacenter** Identify machines that do not work properly (e.g. processing load over network traffic)

**Network traffic** Identify unusual traffic patterns (e.g. traffic amount or origin)
Anomaly detection

Problem statement
Anomaly detection

Problem statement
Anomaly detection

Problem statement
Anomaly detection

Anomaly detection algorithm

1. Choose features that are indicative of anomalous examples

In the Gaussian distribution, fit the parameters:

\[ \mu_j = \frac{1}{n} \sum_{i=1}^{n} x_i^j \] (mean of a Gaussian distribution)

\[ \sigma_j^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i^j - \mu_j)^2 \] (standard deviation of a Gaussian distribution)

For a new sample \( x \), compute:

\[ P[x] = \prod_{j=1}^{m} P[x_j; \mu_j, \sigma_j^2] = \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}} \]

Gaussian distribution

anomaly if \( P[x] < \varepsilon \)
Anomaly detection

Anomaly detection algorithm

1. Choose features that are indicative of anomalous examples
2. Conditioning on a Gaussian distribution, fit the parameters

\[ \mu_j = \frac{1}{n} \sum_{i=1}^{n} x_j^{(i)} \] (mean of a Gaussian distribution)

\[ \sigma_j^2 = \frac{1}{n} \sum_{i=1}^{n} \left( x_j^{(i)} - \mu_j \right)^2 \] (standard deviation of a Gaussian distribution)
Anomaly detection

Anomaly detection algorithm

1. Choose features that are indicative of anomalous examples
2. Conditioning on a Gaussian distribution, fit the parameters
   \[
   \mu_j = \frac{1}{n} \sum_{i=1}^{n} x_j^{(i)} \quad \text{(mean of a Gaussian distribution)}
   \]
   \[
   \sigma_j^2 = \frac{1}{n} \sum_{i=1}^{n} (x_j^{(i)} - \mu_j)^2 \quad \text{(standard deviation of a Gaussian distribution)}
   \]
3. For a new sample \( x \), compute
   \[
   P[x] = \prod_{j=1}^{m} P[x_j; \mu_j, \sigma_j^2] = \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi\sigma_j}} e^{-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}}
   \]
   Gaussian distribution
Anomaly detection

Anomaly detection algorithm

1. Choose features that are indicative of anomalous examples
2. Conditioning on a Gaussian distribution, fit the parameters
   \[
   \mu_j = \frac{1}{n} \sum_{i=1}^{n} x_j^{(i)} \quad \text{(mean of a Gaussian distribution)}
   \]
   \[
   \sigma_j^2 = \frac{1}{n} \sum_{i=1}^{n} \left( x_j^{(i)} - \mu_j \right)^2 \quad \text{(standard deviation of a Gaussian distribution)}
   \]
3. For a new sample \( x \), compute
   \[
   P[x] = \prod_{j=1}^{m} P[x_j; \mu_j, \sigma_j^2] = \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi\sigma_j}} e^{-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}}
   \]
4. anomaly if \( P[x] < \varepsilon \)
Anomaly detection

Example
Anomaly detection

Example
Anomaly detection

Example
Anomaly detection

Example
Anomaly detection

Problem statement

Choice of good values for $\varepsilon$

Using crossvalidation and testing sets, calculate

- Precision/Recall
- $F_1$-score
- ...
In anomaly detection, we have so far assumed Gaussian distributed features.
In anomaly detection, we have so far assumed Gaussian distributed features.

→ What if the feature distribution is not Gaussian?
Anomaly detection

Generate new features with a more Gaussian-like distribution

Original Data

Data²

Data¹.⁴
Anomaly detection

Non-Gaussian features

Possible operations on features

\[ x_{\text{new}} = \log(x) \]
\[ x_{\text{new}} = \sqrt{x} \]
\[ x_{\text{new}} = x^{\frac{1}{3}} \]
\[ x_{\text{new}} = \log(x + k) \]

\[ \vdots \]
Note that there are cases in which the anomaly looks perfectly normal when considering each dimension separately.
Anomaly detection

Multivariate Gaussian Distribution

- Note that there are cases in which the anomaly looks perfectly normal when considering each dimension separately

→ The consideration of multivariate Gaussian distributions might be suggestive in order to detect such anomalies.
Outline

Anomaly detection

Recommender systems

Stochastic Classification

Online learning
Recommender systems
Recommender systems
Recommender systems
Recommender systems
Recommender systems
Recommender systems
## Recommender systems

<table>
<thead>
<tr>
<th>Item</th>
<th>Rating</th>
<th>Rating</th>
<th>Rating</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Item" /></td>
<td><img src="rating1" alt="Rating" /></td>
<td><img src="rating2" alt="Rating" /></td>
<td><img src="rating3" alt="Rating" /></td>
<td><img src="rating4" alt="Rating" /></td>
</tr>
<tr>
<td><img src="image2" alt="Item" /></td>
<td><img src="rating1" alt="Rating" /></td>
<td><img src="rating2" alt="Rating" /></td>
<td><img src="rating3" alt="Rating" /></td>
<td><img src="rating4" alt="Rating" /></td>
</tr>
<tr>
<td><img src="image3" alt="Item" /></td>
<td><img src="rating1" alt="Rating" /></td>
<td><img src="rating2" alt="Rating" /></td>
<td><img src="rating3" alt="Rating" /></td>
<td><img src="rating4" alt="Rating" /></td>
</tr>
<tr>
<td><img src="image4" alt="Item" /></td>
<td><img src="rating1" alt="Rating" /></td>
<td><img src="rating2" alt="Rating" /></td>
<td><img src="rating3" alt="Rating" /></td>
<td><img src="rating4" alt="Rating" /></td>
</tr>
<tr>
<td><img src="image5" alt="Item" /></td>
<td><img src="rating1" alt="Rating" /></td>
<td><img src="rating2" alt="Rating" /></td>
<td><img src="rating3" alt="Rating" /></td>
<td><img src="rating4" alt="Rating" /></td>
</tr>
<tr>
<td><img src="image6" alt="Item" /></td>
<td><img src="rating1" alt="Rating" /></td>
<td><img src="rating2" alt="Rating" /></td>
<td><img src="rating3" alt="Rating" /></td>
<td><img src="rating4" alt="Rating" /></td>
</tr>
<tr>
<td><img src="image7" alt="Item" /></td>
<td><img src="rating1" alt="Rating" /></td>
<td><img src="rating2" alt="Rating" /></td>
<td><img src="rating3" alt="Rating" /></td>
<td><img src="rating4" alt="Rating" /></td>
</tr>
</tbody>
</table>
Recommender systems

Task/Problem of Recommender systems

Given these ratings for a number of products, predict how the user-ratings for products that have not yet been rated
<table>
<thead>
<tr>
<th>Tools</th>
<th>Accessories</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>0.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3</td>
</tr>
<tr>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Tools</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
</tr>
<tr>
<td>Anomaly detection</td>
<td></td>
</tr>
<tr>
<td>Recommender systems</td>
<td>0.9</td>
</tr>
<tr>
<td>Stochastic Classification</td>
<td>1.0</td>
</tr>
<tr>
<td>Online learning</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
</tr>
</tbody>
</table>

Machine Learning and Pervasive Computing
Represent items as feature vectors

\[ x^{(4)} = \begin{bmatrix} 0.2 \\ 0.9 \end{bmatrix} \quad W^{(1)} = \begin{bmatrix} 5 \\ 1 \end{bmatrix} \]

\[ \Rightarrow (W^{(1)})^T x^{(4)} = 0.2 \cdot 5 + 0.9 \cdot 1 = 1.9 \]
Learn weights from provided ratings for single user $j$ (Linear regression):

$$\min \frac{1}{W(j)} \sum_{i: y^{(i,j)} \neq ?} \left( \left( W(j) \right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{F} \left( W_k(j) \right)^2$$
Learn weights from provided ratings for single all users $1, \ldots, N$:

$$\min_{W^{(1)}, \ldots, W^{(N)}} \frac{1}{2} \sum_{j=1}^{N} \sum_{i:y(i,j) \neq ?} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right) + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} (W_k^{(j)})^2$$
Recommender systems

Optimisation algorithm

\[
\min_{W^{(1)}, \ldots, W^{(N)}} \frac{1}{2} \sum_{j=1}^{N} \sum_{i: y(i,j) \neq ?} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} (W^{(j)}_k)^2
\]
Recommender systems

Optimisation algorithm

$$\min_{W^{(1)},...,W^{(N)}} \frac{1}{2} \sum_{j=1}^{N} \sum_{i:y(i,j)\neq?} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} (W^{(j)}_{k})^2$$

Gradient descent update:

$$W^{(j)}_{k} = W^{(j)}_{k} - \alpha \left( \sum_{i:y(i,j)\neq?} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right) x^{(i)}_{k} + \lambda W^{(j)}_{k} \right)$$
Recommender systems

Optimisation algorithm

\[
\min_{W^{(1)}, \ldots, W^{(N)}} \frac{1}{2} \sum_{j=1}^{N} \sum_{i : y(i,j) \neq ?} \left( \left(W^{(j)}\right)^T x^{(i)} - y(i,j) \right) + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} \left(W^{(j)}_k\right)^2
\]

Gradient descent update:

\[
W^{(j)}_k = W^{(j)}_k - \alpha \left( \sum_{i = y(i,j) \neq ?} \left( \left(W^{(j)}\right)^T x^{(i)} - y(i,j) \right) x^{(i)}_k + \lambda W^{(j)}_k \right)
\]

partial derivative
Recommender systems

Collaborative filtering

We are able to calculate the weights given the feature vectors
We are able to calculate the weights given the feature vectors.

→ But how do we obtain these feature vectors?
Recommender systems

Collaborative filtering

Users might tell their preferences

e.g. more interested in tools or accessories

\[
W^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad W^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad W^{(3)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} \quad W^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}
\]
Recommender systems

Collaborative filtering

Users might tell their preferences

e.g. more interested in tools or accessories

\[ W^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad W^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad W^{(3)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} \quad W^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} \]

\[ (W^{(1)})^T x^{(1)} \approx 4; \quad (W^{(2)})^T x^{(1)} \approx 5; \quad (W^{(3)})^T x^{(1)} \approx ?; \quad (W^{(4)})^T x^{(1)} \approx 0 \]
Recommender systems

Collaborative filtering

Users might tell their preferences

e.g. more interested in tools or accessories

\[
W^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad W^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad W^{(3)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} \quad W^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}
\]

\[
(W^{(1)})^T x^{(1)} \approx 4; \quad (W^{(2)})^T x^{(1)} \approx 5;
\]
\[
(W^{(3)})^T x^{(1)} \approx ?; \quad (W^{(4)})^T x^{(1)} \approx 0
\]

From these weights we can estimate the feature values
Recommender systems
Collaborative filtering

Optimisation algorithm
Given the weights/preferences $W^{(1)}, \ldots, W^{(N)}$, we are able to infer a feature $x^{(i)}$

$$
\min_{x^{(i)}} \frac{1}{2} \sum_{j:y^{(i,j)}\neq?} \left( \left( W^{(j)} \right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{F} \left( x^{(i)}_k \right)^2
$$
Recommender systems

Collaborative filtering

Optimisation algorithm
Given the weights/preferences $W^{(1)}, \ldots, W^{(N)}$, we are able to infer a feature $x^{(i)}$

$$
\min_{x^{(i)}} \frac{1}{2} \sum_{j: y(i,j) \neq ?} \left( (W^{(j)})^T x^{(i)} - y(i,j) \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{F} (x_k^{(j)})^2
$$

Given the weights/preferences $W^{(1)}, \ldots, W^{(N)}$, we are able to infer $x^{(1)}, \ldots, x^{(n)}$

$$
\min_{x^{(1)}, \ldots, x^{(n)}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j: y(i,j) \neq ?} \left( (W^{(j)})^T x^{(i)} - y(i,j) \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{F} (x_k^{(i)})^2
$$
Recommender systems

Collaborative filtering

Naive (iterative) Collaborative filtering algorithm

Given $x^{(1)}, \ldots, x^{(n)}$, we are able to estimate $W^{(1)}, \ldots, W^{(N)}$

Given $W^{(1)}, \ldots, W^{(N)}$, we are able to estimate $x^{(1)}, \ldots, x^{(n)}$
Recommender systems

Collaborative filtering

Naive (iterative) Collaborative filtering algorithm

Given $x^{(1)}, \ldots, x^{(n)}$, we are able to estimate $W^{(1)}, \ldots, W^{(N)}$.

Given $W^{(1)}, \ldots, W^{(N)}$, we are able to estimate $x^{(1)}, \ldots, x^{(n)}$.

Collaborative filtering (naive)

Init: Randomly initialise the $W^{(i)}$

Repeat:  
- Estimate the $x^{(i)}$ from the $W^{(i)}$
- Estimate the $W^{(i)}$ from the $x^{(i)}$
Recommender systems

Collaborative filtering

Naive (iterative) Collaborative filtering algorithm

Given $x^{(1)}, \ldots, x^{(n)}$, we are able to estimate $W^{(1)}, \ldots, W^{(N)}$

Given $W^{(1)}, \ldots, W^{(N)}$, we are able to estimate $x^{(1)}, \ldots, x^{(n)}$

Collaborative filtering (naive)

Init: Randomly initialise the $W^{(i)}$

Repeat: 
- Estimate the $x^{(i)}$ from the $W^{(i)}$
- Estimate the $W^{(i)}$ from the $x^{(i)}$

→ CF iteratively improves the estimates for $x^{(i)}$ and $W^{(i)}$

→ Algorithm collaborates with users: by providing some information about their preferences, it computes and improves the features
Recommender systems

Collaborative filtering

\[
\min_{W^{(1)}, \ldots, W^{(N)}} \frac{1}{2} \sum_{j=1}^{N} \sum_{i: y(i,j) \neq \cdot} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} \left( W^{(j)}_k \right)^2
\]

\[
\min_{x^{(1)}, \ldots, x^{(n)}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j: y(i,j) \neq \cdot} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{F} \left( x^{(i)}_k \right)^2
\]
Recommender systems

Collaborative filtering

\[
\begin{align*}
\min_{W^{(1)}, \ldots, W^{(N)}} & \quad \frac{1}{2} \sum_{j=1}^{N} \sum_{i : y(i,j) \neq ?} \left( (W^{(j)})^T x(i) - y(i,j) \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} (W^{(j)}_k)^2 \\
\min_{x^{(1)}, \ldots, x^{(n)}} & \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{j : y(i,j) \neq ?} \left( (W^{(j)})^T x(i) - y(i,j) \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{F} (x^{(i)}_k)^2
\end{align*}
\]
Recommender systems

Collaborative filtering

\[
\begin{align*}
\min_{W^{(1)}, \ldots, W^{(N)}} & \quad \frac{1}{2} \sum_{j=1}^{N} \sum_{i:y(i,j) \neq ?} \left((W^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} \left(W^{(j)}_k\right)^2 \\
\min_{x^{(1)}, \ldots, x^{(n)}} & \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{j:y(i,j) \neq ?} \left((W^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{F} \left(x^{(i)}_k\right)^2
\end{align*}
\]

Minimize \(W^{(i)}\) and \(x^{(i)}\) simultaneously:

\[
\begin{align*}
\min_{x^{(1)}, \ldots, x^{(n)}, W^{(1)}, \ldots, W^{(N)}} & \quad \frac{1}{2} \sum_{i,j:y(i,j) \neq ?} \left((W^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 \\
& \quad + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{F} \left(x^{(i)}_k\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{F} \left(W^{(j)}_k\right)^2
\end{align*}
\]
Recommender systems

Collaborative filtering

Collaborative filtering algorithm

**Init:** Randomly initialise the $W^{(j)}$ and $x^{(i)}$

**Optimisation:** Simultaneously minimise the above function for $W^{(j)}$ and $x^{(i)}$

**Gradient descent:**

\[
 x_k^{(i)} = x_k^{(i)} - \alpha \left( \sum_{j=y(i,j) \neq ?} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right) W_k^{(j)} + \lambda x_k^{(i)} \right)
\]

\[
 W_k^{(j)} = W_k^{(j)} - \alpha \left( \sum_{i=y(i,j) \neq ?} \left( (W^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda W_k^{(j)} \right)
\]

**Prediction:** For a user $i$ with parameters $W^{(j)}$ and an item with learned features $x$, estimate a rating of $(W^{(j)})^T x$
Outline

Anomaly detection

Recommender systems

Stochastic Classification

Online learning
Stochastic Classification

Classification algorithm typically become very slow when data size increases.
Classification algorithm typically become very slow when data size increases

→ This is because they loop repeatedly over the complete data set until convergence
Stochastic Classification

Classification algorithm typically become very slow when data size increases

→ This is because they loop repeatedly over the complete data set until convergence

Solution

- Randomly iterate the update only over individual items instead of repeatedly considering the whole data set.
Stochastic Classification

Example: Gradient descent

\[
\text{minimize } E[W] = \frac{1}{2n} \sum_{i=1}^{n} \left( W^T x^{(i)} - y^{(i)} \right)^2
\]

Repeat \( \forall j \) : 
\[
W_j = W_j - \delta \cdot \frac{\partial}{\partial W_j} E[W_j]
\]

\( \rightarrow \forall j \) : 
\[
W_j = W_j - \delta \cdot \frac{1}{n} \sum_{i=1}^{n} \left( W_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)}
\]
Stochastic Classification

Example: Gradient descent

For single gradient descent-step, algorithms loops over all samples!

\[
\text{minimize } E[W] = \frac{1}{2n} \sum_{i=1}^{n} \left( W^T x^{(i)} - y^{(i)} \right)^2
\]

Repeat \( \forall j : W_j = W_j - \delta \cdot \frac{\partial}{\partial W_j} E[W_j] \)

\( \rightarrow \forall j : W_j = W_j - \delta \cdot \frac{1}{n} \sum_{i=1}^{n} \left( W_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)} \)
Stochastic Classification

Example: Gradient descent

→ For a single gradient descent-step, the algorithm loops over all samples!

\[
\text{minimize } E[W] = \frac{1}{2n} \sum_{i=1}^{n} \left( W^T x^{(i)} - y^{(i)} \right)^2
\]

Repeat \( \forall j : W_j = W_j - \delta \cdot \frac{\partial}{\partial W_j} E[W_j] \)

→ \( \forall j : W_j = W_j - \delta \cdot \frac{1}{n} \sum_{i=1}^{n} \left( W_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)} \)
Stochastic Classification
Example: Gradient descent

→ For a single gradient descent-step, the algorithm loops over all samples!

To speed up the algorithm, compute gradient descent updates from individual training samples (randomly ordered)

minimize \( E[W] = \frac{1}{2n} \sum_{i=1}^{n} \left( W^T x^{(i)} - y^{(i)} \right)^2 \)

Repeat \( \forall j : W_j = W_j - \delta \cdot \frac{\partial}{\partial W_j} E[W_j] \)

→ \( \forall j : W_j = W_j - \delta \cdot \frac{1}{n} \sum_{i=1}^{n} \left( W_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)} \)
Stochastic Classification

Example: Gradient descent

**Standard:**

\[
\text{minimize } E[W] = \frac{1}{2n} \sum_{i=1}^{n} \left( W^T x^{(i)} - y^{(i)} \right)^2
\]

Repeat \( \forall j : W_j = W_j - \delta \cdot \frac{\partial}{\partial W_j} E[W_j] \)

\[
\rightarrow \forall j : W_j = W_j - \delta \cdot \frac{1}{n} \sum_{i=1}^{n} \left( W_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)}
\]
Stochastic Classification

Example: Gradient descent

**Standard:**

\[
\text{minimize } E[W] = \frac{1}{2n} \sum_{i=1}^{n} \left( W^T x^{(i)} - y^{(i)} \right)^2
\]

Repeat \( \forall j : W_j = W_j - \delta \cdot \frac{\partial}{\partial W_j} E[W_j] \)

\[
\rightarrow \forall j : W_j = W_j - \delta \cdot \frac{1}{n} \sum_{i=1}^{n} \left( W_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)}
\]

**Stochastic:**

Repeat over all training examples \( i \) (random order):

\[
\Rightarrow \forall j : \ W_j = W_j - \delta \cdot \left( W_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)}
\]
The stochastic implementation will generally move the parameters towards the global minimum (...but not always!)
The stochastic implementation will generally move the parameters towards the global minimum (*but not always!*).

Greatly speeds up the algorithm as it does not loop over all samples in each single iteration.
The stochastic implementation will generally move the parameters towards the global minimum (...but not always!)

Greatly speeds up the algorithm as it does not loop over all samples in each single iteration

Tradeoff use $1 \leq k \leq n$ random examples for each gradient descent update

$l = 1, 1 + k, 1 + 2k, \ldots$

$$\Rightarrow \forall j : W_j = W_j - \delta \cdot \frac{1}{l} \sum_{l=i}^{i+k} \left(W_j x_j^{(i)} - y^{(i)}\right) \cdot x_j^{(i)}$$
The stochastic implementation will generally move the parameters towards the global minimum (...but not always!)

Greatly speeds up the algorithm as it does not loop over all samples in each single iteration

Tradeoff: use \(1 \leq k \leq n\) random examples for each gradient descent update

\[ l = 1, 1+k, 1+2k, \ldots \]

\[ \Rightarrow \forall j : W_j = W_j - \delta \cdot \frac{1}{l} \sum_{l=i}^{i+k} (W_j x_j^{(i)} - y^{(i)}) \cdot x_j^{(i)} \]

\(k > 1\) might be faster than \(k = 1\) for parallelized code.
Outline

Anomaly detection

Recommender systems

Stochastic Classification

Online learning
Online learning

Problem formulation

Online learning

Given a continuous stream of input data, update the parameters of your algorithm on-the-fly

Example: learn user behaviour from website users
Online learning

Problem formulation

Given a continuous stream of input data, update the parameters of your algorithm on-the-fly.

**Example:** learn user behaviour from website users.

**Similar to stochastic classification:** Update the parameters based on individual training examples.

\[ \forall j : \quad W_j = W_j - \delta \cdot (h(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \]
Online learning

Problem formulation

Given a continuous stream of input data, update the parameters of your algorithm on-the-fly

**Example:** learn user behaviour from website users

**Similar to stochastic classification:** Update the parameters based on individual training examples

\[ \forall j : W_j = W_j - \delta \cdot (h(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \]

⇒ Able to adapt to changing user behaviour over time
Outline

Anomaly detection

Recommender systems

Stochastic Classification

Online learning
Questions?

Stephan Sigg
stephan.sigg@cs.uni-goettingen.de
Literature