Aalto University
School of Electrical Engineering

CS-C3240 – Machine Learning D

Anomaly detection, Recommender Systems, Online learning

Stephan Sigg

Department of Communications and Networking
Aalto University, School of Electrical Engineering
stephan.sigg@aalto.fi

Version 1.0, January 19, 2022

Learning goals

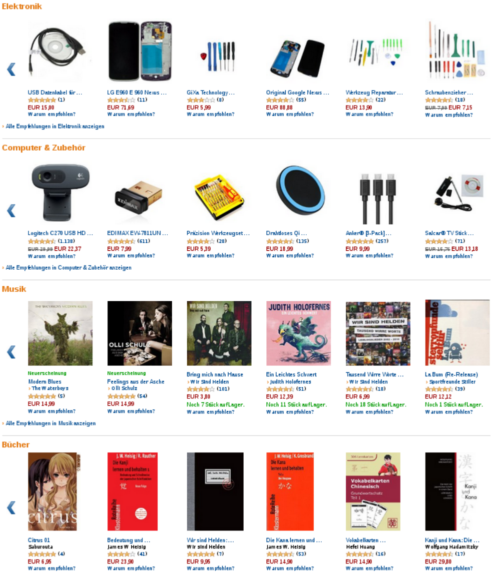
- Collaborative Filtering
- Recommender Systems
- Batch gradient descent
- Online learning

Outline

Recommender systems

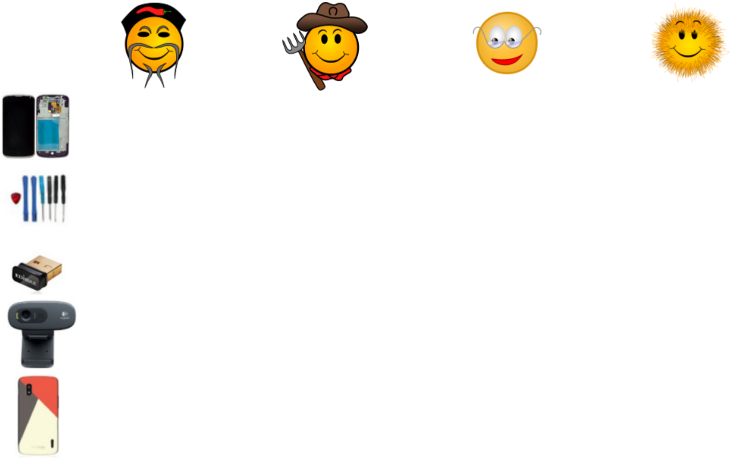
Stochastic Classification

Online learning

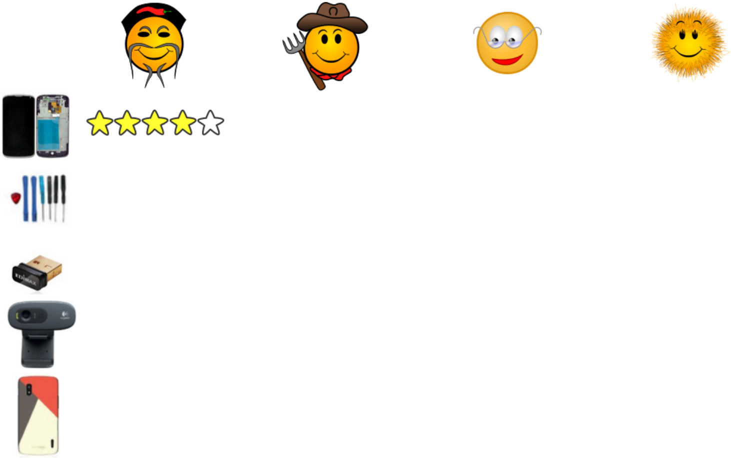


Elektronik
USB Datenkabel für …
(1)
EUR 15,00
Warum empfohlen?
LG E960 E 960 Nexus …
(11)
EUR 71,69
Warum empfohlen?
GiXa Technology …
(8)
EUR 5,99
Warum empfohlen?
Original Google Nexus …
(55)
EUR 88,88
Warum empfohlen?
Werkzeug Reparatur …
(22)
EUR 13,90
Warum empfohlen?
Schraubenzieher …
(18)
EUR 7,99 EUR 7,15
Warum empfohlen?
› Alle Empfehlungen in Elektronik anzeigen
Computer & Zubehör
Logitech C270 USB HD …
(1.138)
EUR 29,99 EUR 22,37
Warum empfohlen?
EDIMAX EW-7811UN …
(611)
EUR 7,99
Warum empfohlen?
Präzision Werkzeugset …
(28)
EUR 5,39
Warum empfohlen?
Drahtloses Qi …
(135)
EUR 18,99
Warum empfohlen?
Anker® [3-Pack] …
(257)
EUR 9,99
Warum empfohlen?
Salcar® TV Stick …
(71)
EUR 15,76 EUR 13,18
Warum empfohlen?
› Alle Empfehlungen in Computer & Zubehör anzeigen
Musik
Neuerscheinung
Modern Blues
› The Waterboys
(5)
EUR 14,99
Warum empfohlen?
Neuerscheinung
Feelings aus der Asche
› Olli Schulz
(54)
EUR 14,99
Warum empfohlen?
Bring mich nach Hause
› Wir Sind Helden
(101)
EUR 3,80
Noch 7 Stück auf Lager.
Warum empfohlen?
Ein Leichtes Schwert
› Judith Holofernes
(51)
EUR 12,39
Noch 11 Stück auf Lager.
Warum empfohlen?
Tausend Wirre Worte …
› Wir Sind Helden
(18)
EUR 6,99
Noch 18 Stück auf Lager.
Warum empfohlen?
La Bum (Re-Release)
› Sportfreunde Stiller
(39)
EUR 12,12
Noch 1 Stück auf Lager.
Warum empfohlen?
› Alle Empfehlungen in Musik anzeigen
Bücher
Citrus 01
Saburouta
(4)
EUR 6,95
Warum empfohlen?
Bedeutung und …
James W Heisig
(41)
EUR 23,90
Warum empfohlen?
Wir sind Helden: …
Wir sind Helden
(7)
EUR 9,95
Warum empfohlen?
Die Kana lernen und …
James W. Heisig
(53)
EUR 14,90
Warum empfohlen?
Vokabelkarten …
Hefei Huang
(16)
EUR 14,90
Warum empfohlen?
Kanji und Kana: Die …
Wolfgang Hadamitzky
(17)
EUR 29,80
Warum empfohlen?

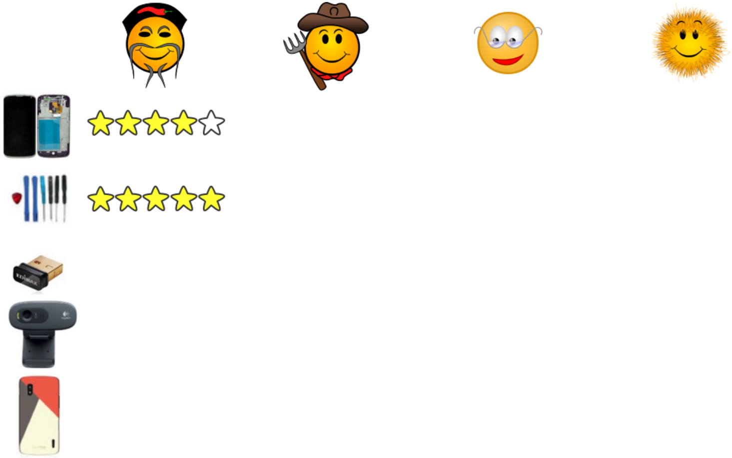
Recommender systems



Recommender systems



Recommender systems

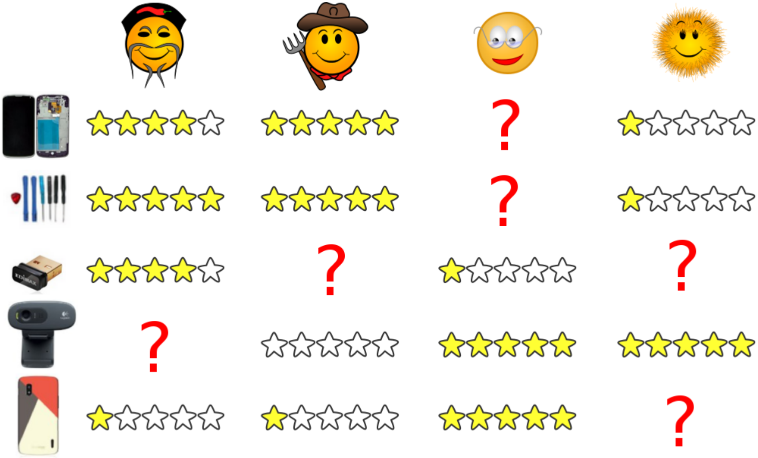


Recommender systems

Recommender systems

Recommender systems

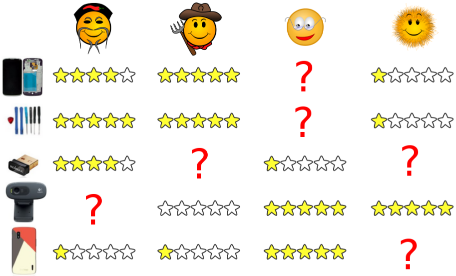
Recommender systems

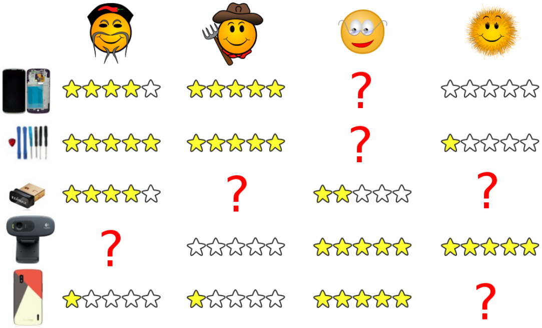


Recommender systems

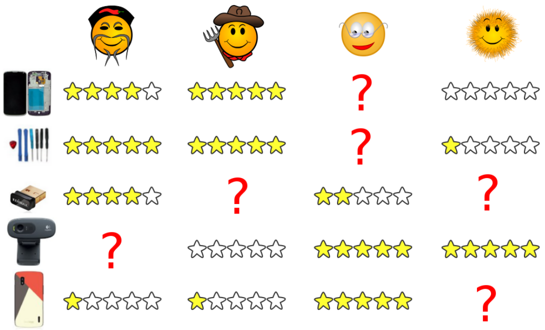
Task of Recommender systems

Given these ratings for a number of products, predict likely user-ratings for products that have not yet been rated

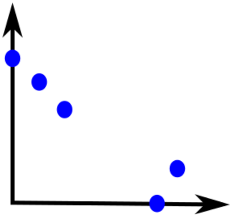


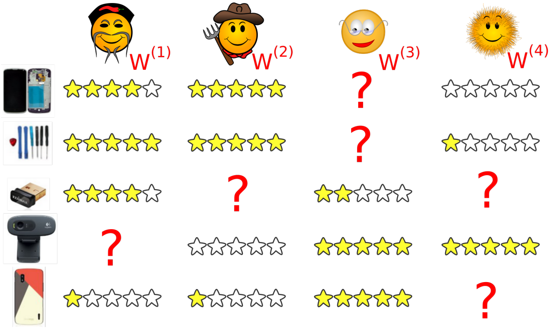


tools	accessories
0.9	0.0
1.0	0.3
0.4	0.8
0.2	0.9
0.0	1.0

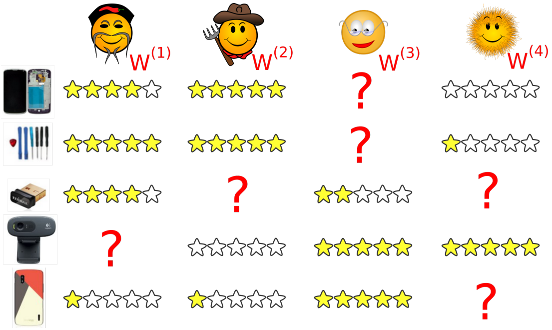


tools	accessories
0.9	0.0
1.0	0.3
0.4	0.8
0.2	0.9
0.0	1.0





tools		accessories
0.9	$x^{(1)}$	0.0
1.0	$x^{(2)}$	0.3
0.4	$x^{(3)}$	0.8
0.2	$x^{(4)}$	0.9
0.0	$x^{(5)}$	1.0

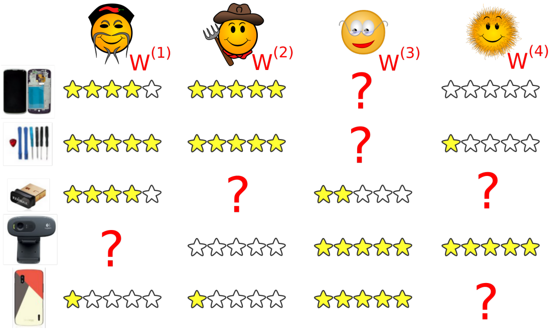


	tools		accessories
	0.9	$x^{(1)}$	0.0
	1.0	$x^{(2)}$	0.3
	0.4	$x^{(3)}$	0.8
	0.2	$x^{(4)}$	0.9
	0.0	$x^{(5)}$	1.0

Represent items in terms of weighted feature vectors

$$x^{(4)} = \begin{bmatrix} 0.2 \\ 0.9 \end{bmatrix} \quad w^{(1)} = \begin{bmatrix} \mathbf{5} \\ \mathbf{1} \end{bmatrix}$$

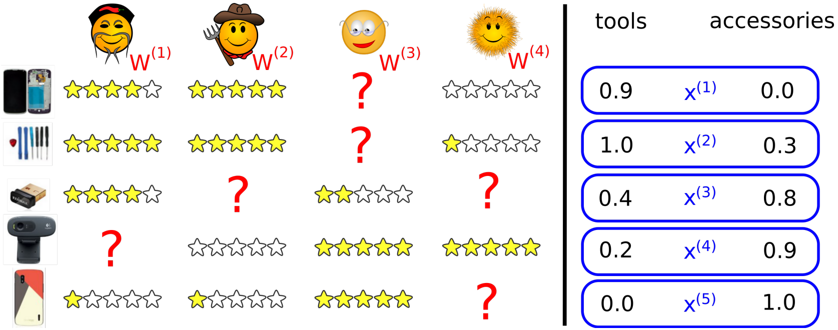
$$h(x) \Rightarrow \left(w^{(1)}\right)^T x^{(4)} = 0.2 \cdot 5 + 0.9 \cdot 1 = 1,9$$



	tools		accessories
	0.9	$x^{(1)}$	0.0
	1.0	$x^{(2)}$	0.3
	0.4	$x^{(3)}$	0.8
	0.2	$x^{(4)}$	0.9
	0.0	$x^{(5)}$	1.0

Learn weights from provided ratings for single user *j* (Logistic regression):

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



Learn weights from provided ratings for all users $1, \ldots, N$:

$$\min_{w^{(1)},\ldots,w^{(N)}} \frac{1}{2}\sum_{j=1}^{N}\sum_{i:y^{(i,j)}\neq ?}\left(\left(w^{(j)}\right)^{T}x^{(i)}-y^{(i,j)}\right)^{2}+\frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{m}\left(w_k^{(j)}\right)^{2}$$

Recommender systems

Optimisation algorithm

$$\min_{w^{(1)},\ldots,w^{(N)}} \frac{1}{2}\sum_{j=1}^{N}\sum_{i:y^{(i,j)}\neq ?}\left(\left(w^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{m}\left(w_k^{(j)}\right)^2$$

Recommender systems

Optimisation algorithm

$$\min_{w^{(1)},\ldots,w^{(N)}} \frac{1}{2}\sum_{j=1}^{N}\sum_{i:y^{(i,j)}\neq ?}\left(\left(w^{(j)}\right)^{T}x^{(i)}-y^{(i,j)}\right)^{2}+\frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{m}\left(w_k^{(j)}\right)^{2}$$

Gradient descent update:

$$w_k^{(j)}=w_k^{(j)}-\delta\left(\sum_{\boldsymbol{i}=\boldsymbol{y}^{(\boldsymbol{i},\boldsymbol{j})}\neq ?}\left(\left(\boldsymbol{w}^{(\boldsymbol{j})}\right)^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}-\boldsymbol{y}^{(\boldsymbol{i},\boldsymbol{j})}\right)\boldsymbol{x}_{\boldsymbol{k}}^{(\boldsymbol{i})}+\boldsymbol{\lambda}\boldsymbol{w}_{\boldsymbol{k}}^{(\boldsymbol{j})}\right)$$

Recommender systems

Optimisation algorithm

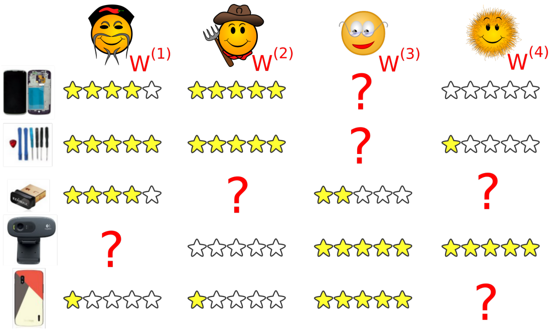
$$\min_{w^{(1)},\dots,w^{(N)}} \frac{1}{2} \sum_{j=1}^{N} \sum_{i:y^{(i,j)}\neq ?} \left(\left(w^{(j)} \right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{N} \sum_{k=1}^{m} \left(w_k^{(j)} \right)^2$$

Gradient descent update:

$$w_k^{(j)} = w_k^{(j)} - \delta \left(\underbrace{\sum_{i=y^{(i,j)}\neq ?} \left(\left(\boldsymbol{w}^{(j)} \right)^T \boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i,j)} \right) \boldsymbol{x}_k^{(i)} + \lambda \boldsymbol{w}_k^{(j)}}_{\text{partial derivative}} \right)$$

Recommender systems

Collaborative filtering

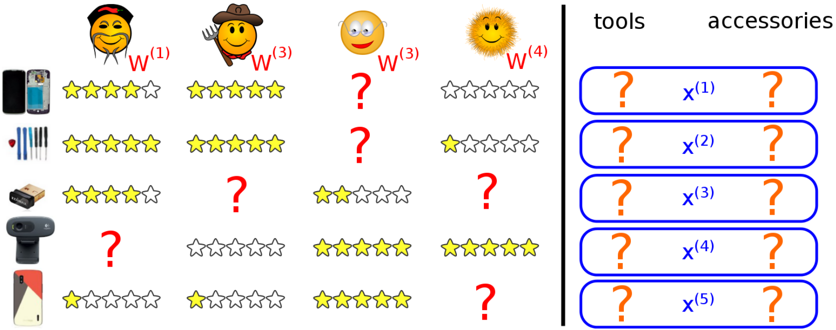


	tools		accessories
	0.9	$x^{(1)}$	0.0
	1.0	$x^{(2)}$	0.3
	0.4	$x^{(3)}$	0.8
	0.2	$x^{(4)}$	0.9
	0.0	$x^{(5)}$	1.0

We are able to calculate the weights given the feature vectors

Recommender systems

Collaborative filtering

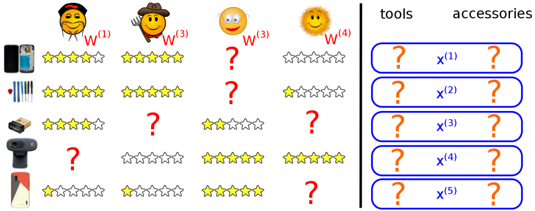


We are able to calculate the weights given the feature vectors

→ But how do we obtain these feature vectors?

Recommender systems

Collaborative filtering



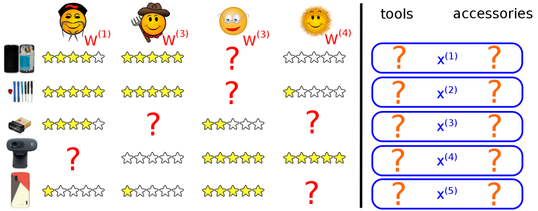
Users might tell their preferences

e.g. more interested in tools or accessories

$$w^{(1)} = \begin{bmatrix} \mathbf{5} \\ \mathbf{0} \end{bmatrix} \quad w^{(2)} = \begin{bmatrix} \mathbf{5} \\ \mathbf{0} \end{bmatrix} \quad w^{(3)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{5} \end{bmatrix} \quad w^{(4)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{5} \end{bmatrix}$$

Recommender systems

Collaborative filtering



Users might tell their preferences

e.g. more interested in tools or accessories

$$w^{(1)} = \begin{bmatrix} \mathbf{5} \\ \mathbf{0} \end{bmatrix} \quad w^{(2)} = \begin{bmatrix} \mathbf{5} \\ \mathbf{0} \end{bmatrix} \quad w^{(3)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{5} \end{bmatrix} \quad w^{(4)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{5} \end{bmatrix}$$

$$\left(w^{(1)}\right)^T x^{(1)} \approx 4; \quad \left(w^{(2)}\right)^T x^{(1)} \approx 5;$$

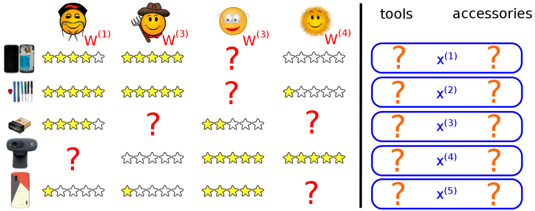
$$\left(w^{(3)}\right)^T x^{(1)} \approx ?; \quad \left(w^{(4)}\right)^T x^{(1)} \approx 0$$

Recommender systems

Collaborative filtering

Users might tell their preferences

e.g. more interested in tools or accessories



$$w^{(1)} = \begin{bmatrix} \mathbf{5} \\ \mathbf{0} \end{bmatrix} \quad w^{(2)} = \begin{bmatrix} \mathbf{5} \\ \mathbf{0} \end{bmatrix} \quad w^{(3)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{5} \end{bmatrix} \quad w^{(4)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{5} \end{bmatrix}$$

$$\left(w^{(1)}\right)^T x^{(1)} \approx 4; \quad \left(w^{(2)}\right)^T x^{(1)} \approx 5;$$

$$\left(w^{(3)}\right)^T x^{(1)} \approx ?; \quad \left(w^{(4)}\right)^T x^{(1)} \approx 0$$

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 $x^{(i)}$

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

Recommender systems

Collaborative filtering

Optimisation algorithm

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

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

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

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

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 $x^{(1)}, \dots, x^{(n)}$, we are able to estimate $w^{(1)}, \dots, w^{(N)}$

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

Collaborative filtering (naive)

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

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

Naive (iterative) Collaborative filtering algorithm

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

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

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

$$\min_{w^{(1)},\dots,w^{(N)}} \quad \frac{1}{2}\sum_{j=1}^{N}\sum_{i:y^{(i,j)}\neq ?}\left(\left(w^{(j)}\right)^{T}x^{(i)} - y^{(i,j)}\right)^{2} + \frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{m}\left(w_k^{(j)}\right)^{2}$$

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

Recommender systems

Collaborative filtering

$$\min_{w^{(1)},\dots,w^{(N)}} \quad \frac{1}{2}\sum_{j=1}^{N}\sum_{i:y^{(i,j)}\neq ?}\left(\left(w^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{m}\left(w_k^{(j)}\right)^2$$

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

Recommender systems

Collaborative filtering

$$\min_{w^{(1)},\dots,w^{(N)}} \quad \frac{1}{2}\sum_{j=1}^{N}\sum_{i:y^{(i,j)}\neq ?}\left(\left(w^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{m}\left(w_k^{(j)}\right)^2$$

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

Minimize $w^{(i)}$ and $x^{(i)}$ simultaneously:

$$\min_{x^{(1)},\dots,x^{(n)},w^{(1)},\dots,w^{(N)}} \quad \frac{1}{2}\sum_{i,j:y^{(i,j)}\neq ?}\left(\left(w^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}\left(x_k^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{N}\sum_{k=1}^{m}\left(w_k^{(j)}\right)^2$$

Recommender systems

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

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

Gradient descent:

$$
\begin{aligned}
x_k^{(i)} &= x_k^{(i)} - \delta \left(\sum_{j=y^{(i,j)}\neq ?} \left(\left(w^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right) w_k^{(j)} + \lambda x_k^{(i)} \right) \\
w_k^{(j)} &= w_k^{(j)} - \delta \left(\sum_{i=y^{(i,j)}\neq ?} \left(\left(w^{(j)}\right)^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda w_k^{(j)} \right)
\end{aligned}
$$

Prediction: For a user i with parameters $w^{(j)}$ and an item with learned featues x, estimate a rating of $\left(w^{(j)}\right)^T x$

Outline

Recommender systems

Stochastic Classification

Online learning

Stochastic Classification

Model training becomes slow with increasing data size

Stochastic Classification

Model training becomes slow with increasing data size

→ Because of repeated looping over the complete data set until convergence

Stochastic Classification

Model training becomes slow with increasing data size

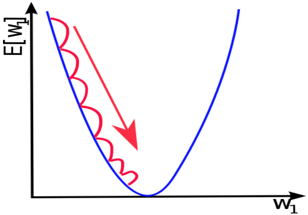
→ Because of repeated looping over the complete data set until convergence

Solution

- Randomly iterate the update only over a subset of items instead of repeatedly considering the whole data set.

Stochastic Classification

Example: Gradient descent



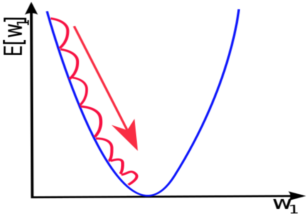
$$\text{minimize } L[(\mathcal{X},\mathcal{Y}),h(\cdot)] = \frac{1}{2n}\sum_{i=1}^{n}\left(w^T x^{(i)} - y^{(i)}\right)^2$$

$$\text{Repeat } \forall j : w_j = w_j - \delta \cdot \frac{\partial L[(\mathcal{X},\mathcal{Y}),h(\cdot)]}{\partial w_j}$$

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

Stochastic Classification

Example: Gradient descent



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

$$\text{minimize } L[(\mathcal{X},\mathcal{Y}),h(\cdot)] = \frac{1}{2n}\sum_{i=1}^{n}\left(w^T x^{(i)} - y^{(i)}\right)^2$$

$$\text{Repeat } \forall j : w_j = w_j - \delta \cdot \frac{\partial L[(\mathcal{X},\mathcal{Y}),h(\cdot)]}{\partial w_j}$$

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

Stochastic Classification

Example: Gradient descent

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

$$
\begin{aligned}
\text{minimize } L[(\mathcal{X},\mathcal{Y}),h(\cdot)] &= \frac{1}{2n}\sum_{i=1}^{n}\left(w^T x^{(i)} - y^{(i)}\right)^2 \\
\text{Repeat } \forall j: w_j &= w_j - \delta \cdot \frac{\partial L[(\mathcal{X},\mathcal{Y}),h(\cdot)]}{\partial w_j} \\
\rightarrow \forall j: w_j &= w_j - \delta \cdot \frac{1}{n}\sum_{i=1}^{n}\left(w_j x_j^{(i)} - y^{(i)}\right) \cdot x_j^{(i)}
\end{aligned}
$$

Stochastic Classification

Example: Gradient descent

$\rightarrow$ For a single gradient descent-step, the algorithms loops over all samples!

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

$$\text{minimize } L[(\mathcal{X},\mathcal{Y}),h(\cdot)] = \frac{1}{2n}\sum_{i=1}^{n}\left(w^T x^{(i)} - y^{(i)}\right)^2$$

$$\text{Repeat } \forall j : w_j = w_j - \delta \cdot \frac{\partial L[(\mathcal{X},\mathcal{Y}),h(\cdot)]}{\partial w_j}$$

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

Stochastic Classification

Example: Gradient descent

Standard:

$$\text{minimize } L[(\mathcal{X},\mathcal{Y}),h(\cdot)] = \frac{1}{2n}\sum_{i=1}^{n}\left(w^T x^{(i)} - y^{(i)}\right)^2$$

$$\text{Repeat } \forall j : w_j = w_j - \delta \cdot \frac{\partial}{\partial w_j} E[w_j]$$

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

Stochastic Classification

Example: Gradient descent

Standard:

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

$$\text{Repeat } \forall j : w_j = w_j - \delta \cdot \frac{\partial}{\partial w_j} E[w_j]$$

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

Stochastic:

Repeat over all training examples i (random order):

$$\Rightarrow \forall j : \quad w_j = w_j - \delta \cdot \left(w_j x_j^{(i)} - y^{(i)}\right) \cdot x_j^{(i)}$$

Stochastic Classification

The stochastic implementation will generally move the parameters towards the global minimum (...but not always!)

Stochastic Classification

The stochastic implementation will generally move the parameters towards the global minimum (...but not always!)

> Greatly speeds up the gradient descent steps as it does not loop over all samples in each single iteration

Stochastic Classification

The stochastic implementation will generally move the parameters towards the global minimum (...but not always!)

Greatly speeds up the gradient descent steps 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

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

$$\Rightarrow \forall j : w_j = w_j - \delta \cdot \frac{1}{l} \sum_{l=i}^{i+k} \left(w_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)}$$

Stochastic Classification

The stochastic implementation will generally move the parameters towards the global minimum (...but not always!)

Greatly speeds up the gradient descent steps 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

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

$$\Rightarrow \forall j : w_j = w_j - \delta \cdot \frac{1}{l} \sum_{l=i}^{i+k} \left(w_j x_j^{(i)} - y^{(i)} \right) \cdot x_j^{(i)}$$

$k > 1$ might be faster than $k = 1$ for parallelized code

Outline

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

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

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

$$\Rightarrow \forall j : \quad w_j = w_j - \delta \cdot \left(h(x^{(i)}) - y^{(i)} \right) \cdot x_j^{(i)}$$

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

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

$$\Rightarrow \forall j: \quad w_j = w_j - \delta \cdot \left(h(x^{(i)}) - y^{(i)}\right) \cdot x_j^{(i)}$$

$\Rightarrow$ Able to adapt to changing user behaviour over time

Questions?

Stephan Sigg
stephan.sigg@aalto.fi

Si Zuo
si.zuo@aalto.fi

Literature

- C.M. Bishop: Pattern recognition and machine learning, Springer, 2007.
- R.O. Duda, P.E. Hart, D.G. Stork: Pattern Classification, Wiley, 2001.