Deep Learning for Recommender Systems

Marie Al Ghossein

June 10, 2024

Crossing Minds

Dauphine | PSL
UNIVERSITÉ PARIS

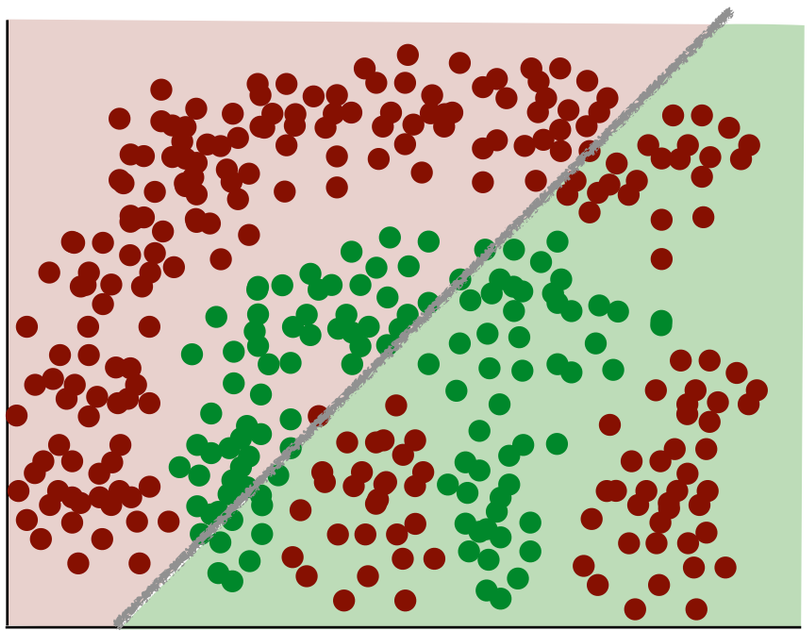
Quick Refresher on Deep Learning

Deep Learning

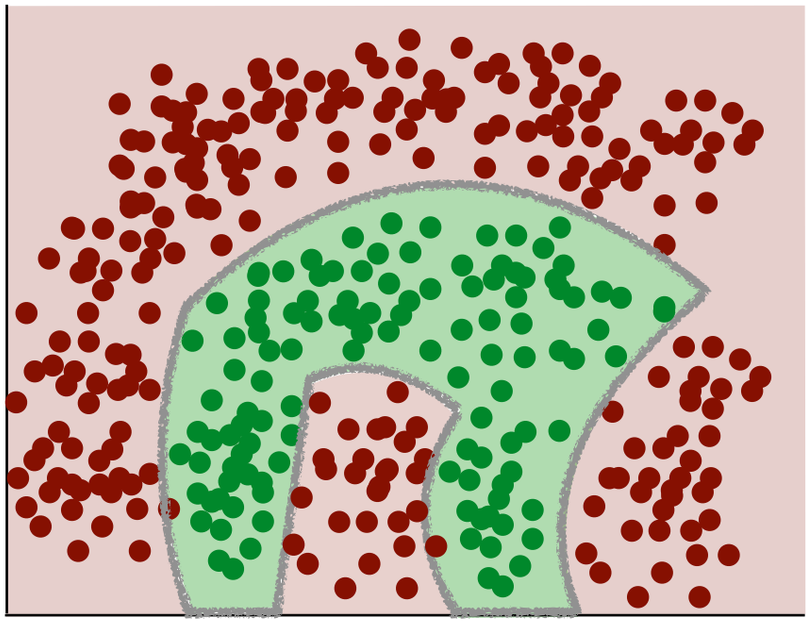
- Class of machine learning algorithms
  - Data processed through a cascade of multiple non-linear layers
  - Layers organized in complex and (usually) deep structures
  - Each layer learns a different representation of the data
  - Each layer builds on the representation learned in the lower layer

Deep Learning - Characteristics

- Learning non-linear models



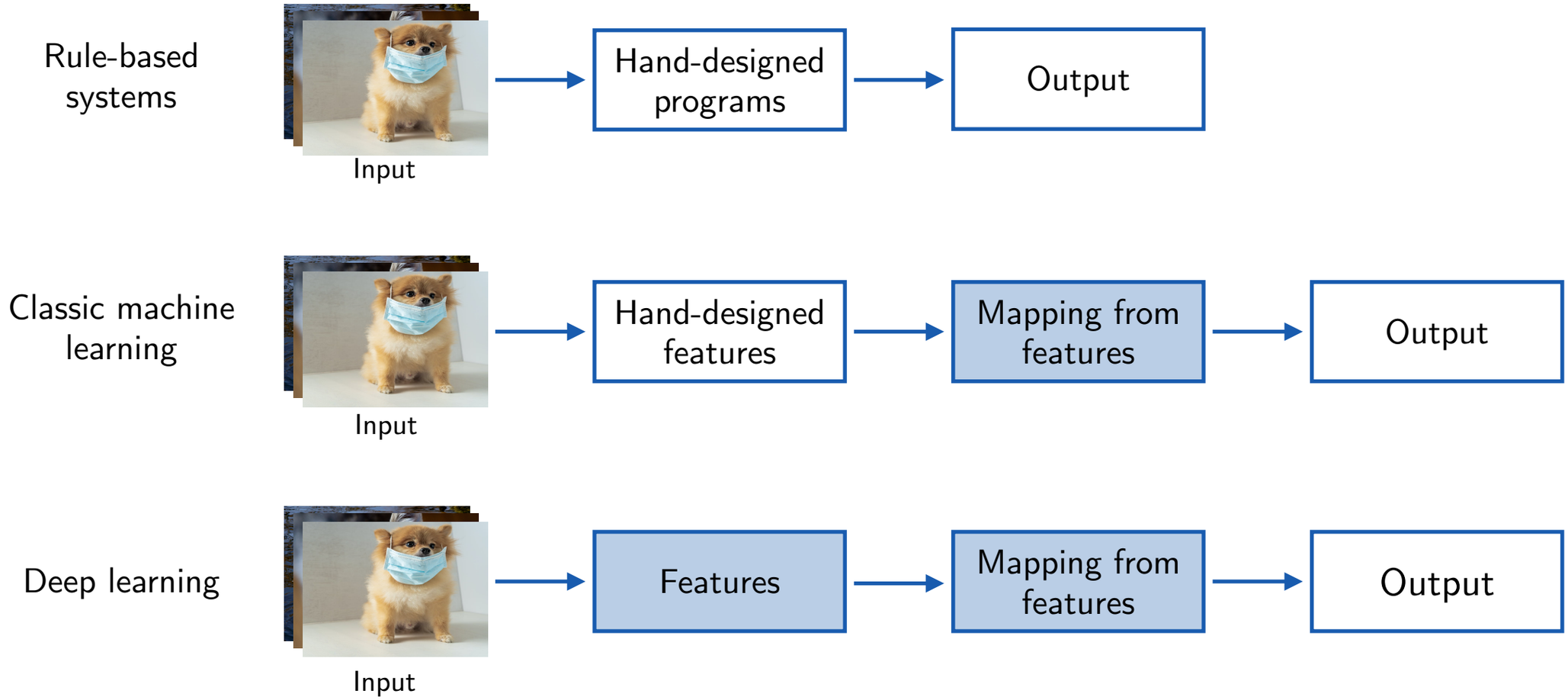
Linear model



Non-linear model

Deep Learning - Characteristics

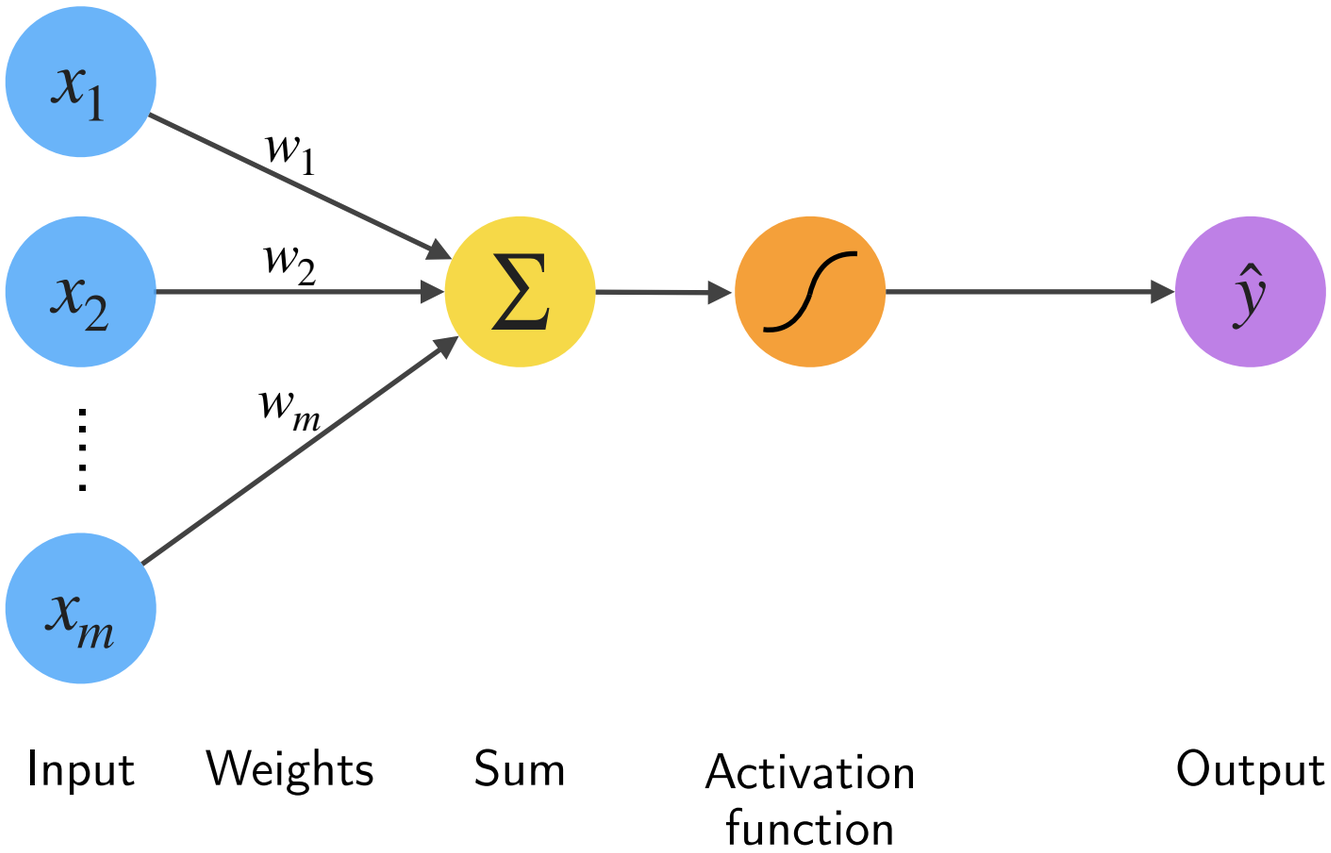
- Learning non-linear models
- Automatic feature extraction from data



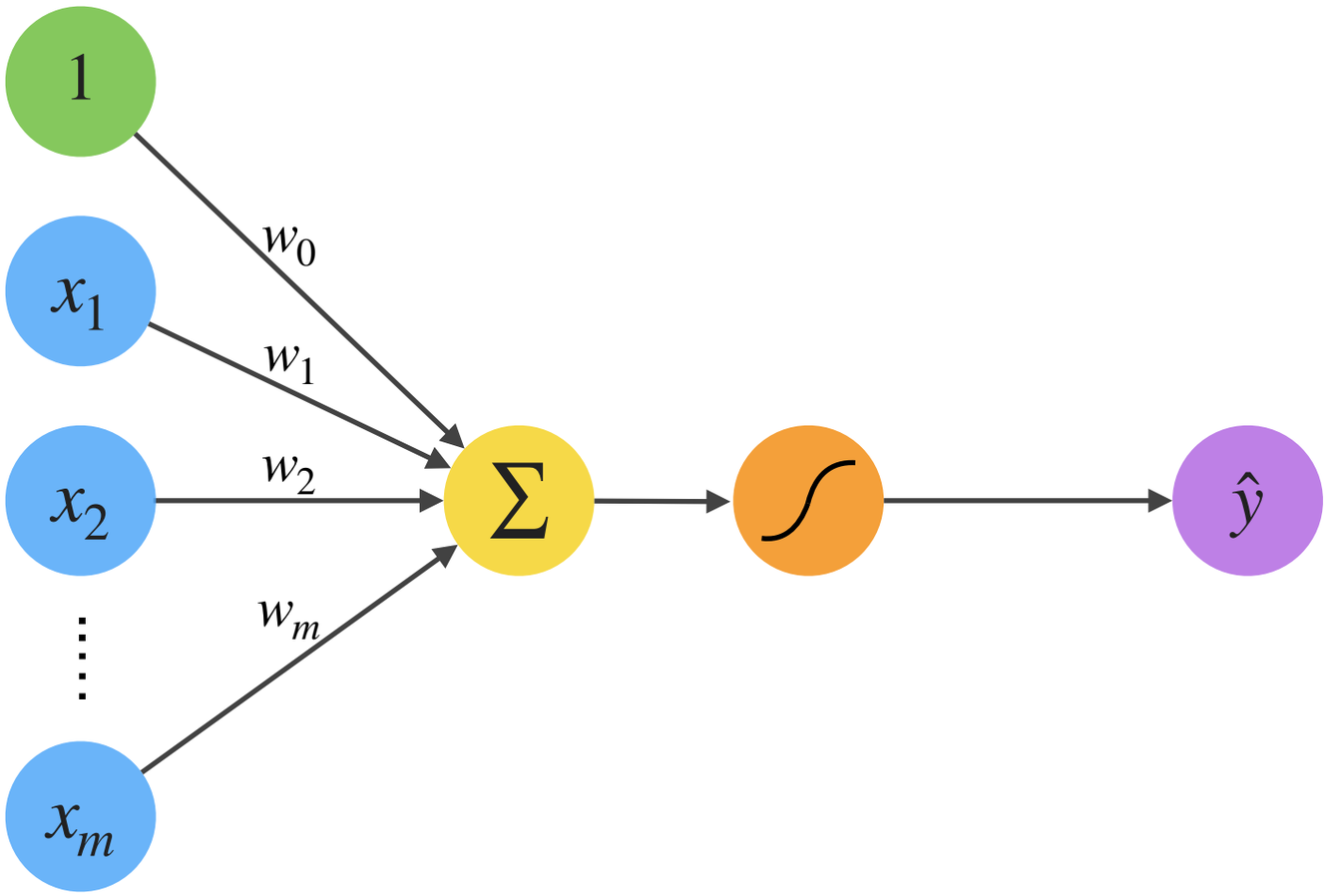
Deep Learning - Characteristics

- Learning non-linear models
- Automatic feature extraction from data
- Feedforward neural networks are universal approximators
  - Ability to approximate any function with arbitrarily low error, if they are "large enough"

The perceptron



The perceptron



Input Weights Sum Activation function Output

$$\mathbf{X} \in \mathbb{R}^m$$

$$\mathbf{W} \in \mathbb{R}^m$$

$$f(x) = \frac{1}{1+e^{-x}}$$

$$\hat{y} = f(w_0 + \mathbf{X}^T\mathbf{W})$$

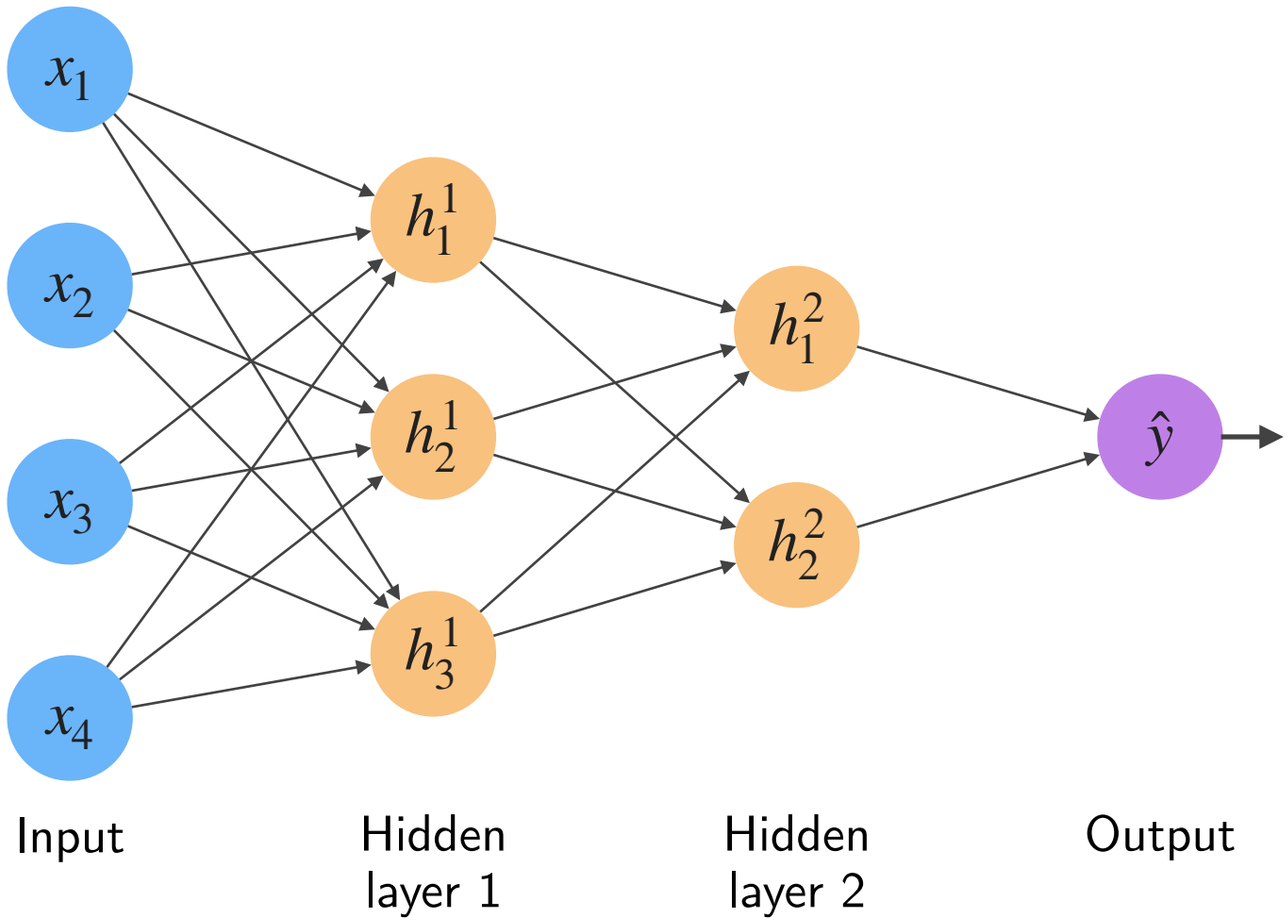
$$= f(w_0 + \sum_i x_i w_i)$$

Neural Networks

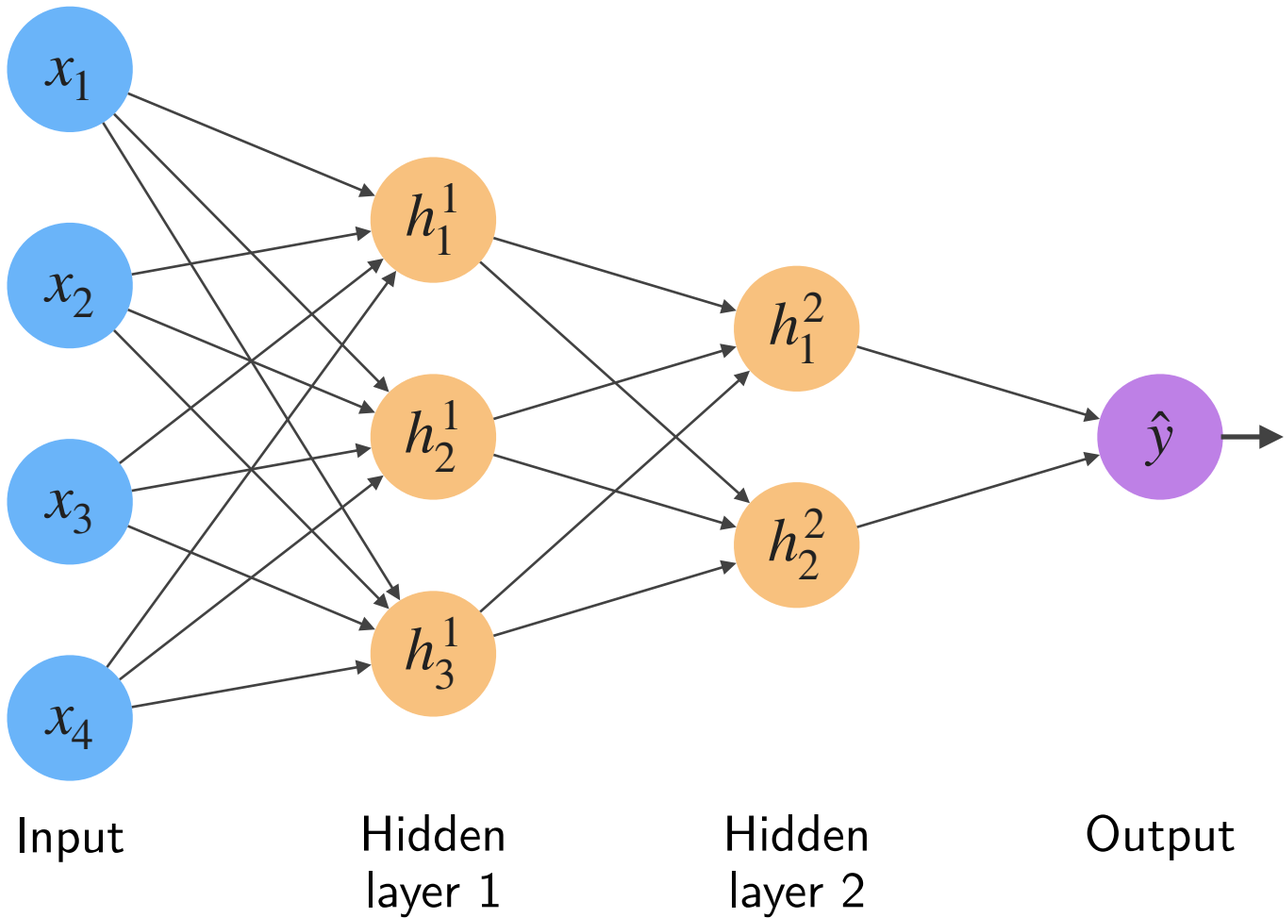
- Artificial neurons connected to each other
  - Connexions done in many different ways
  - Outputs of certain neurons are connected to the input of others

- Most common architectures
  - Feedforward Neural Networks
  - Recurrent Neural Networks
  - Convolutional Neural Networks

Feedforward Neural Networks



Feedforward Neural Networks



- Forward propagation
  - $h_i^0 = x_i$
  - $h_j^k = f(\sum_i w_{i,j}^k \, h_i^{k-1} + b_j)$, $\quad \mathbf{h}^k = f(\mathbf{W}^k \mathbf{h}^{k-1} + \mathbf{b})$
  - $\hat{y} = h^{n+1}$

Training Neural Networks

- Loss optimization

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} J(\mathbf{W})$$

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} L\Big(\phi(x^{(i)}; \mathbf{W}), y^{(i)}\Big)$$

Training Neural Networks

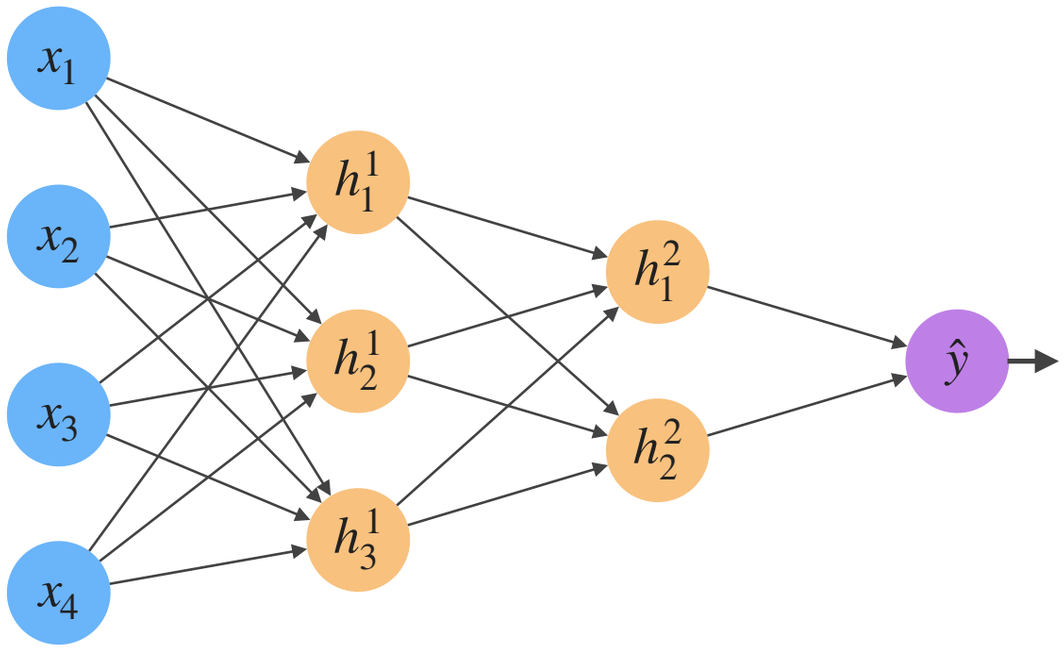
- Loss optimization

$$\mathbf{W}* = \arg\min_{\mathbf{W}} J(\mathbf{W})$$

$$\mathbf{W}* = \arg\min_{\mathbf{W}} \frac{1}{n} \sum_{i=1}^{n} L\Big(\phi(x^{(i)}; \mathbf{W}), y^{(i)}\Big)$$

- Gradient Descent: Algorithm
  - Initialize weights $\mathbf{W}$ randomly
  - Loop until convergence
    - Compute gradient $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
    - Update weights $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
  - Return weights

Backpropagation for computing the gradients



$$J(\mathbf{W}) = E$$

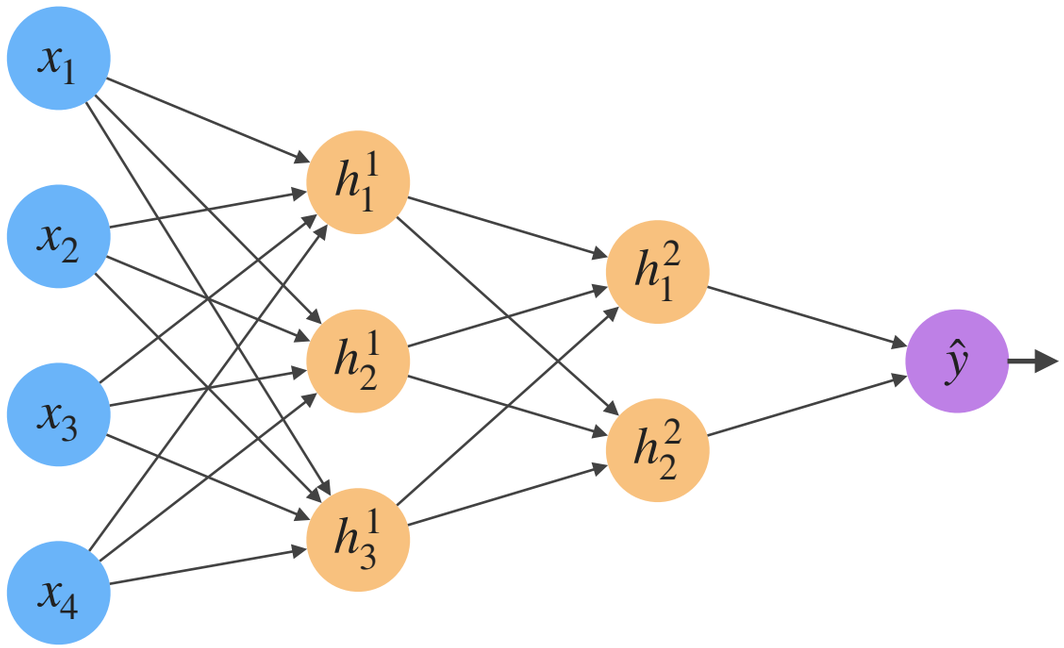
$$h_j^k = f(z_j^k)$$

$$z_j^k = \sum_i w_{i,j}^k \; h_i^{k-1} + b_j$$

Chain rule

$$\frac{\partial E}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; \frac{\partial h_i^k}{\partial z_i^k} \; \frac{\partial z_i^k}{\partial w_{j,i}^k}$$

Backpropagation for computing the gradients



$$J(\mathbf{W}) = E$$

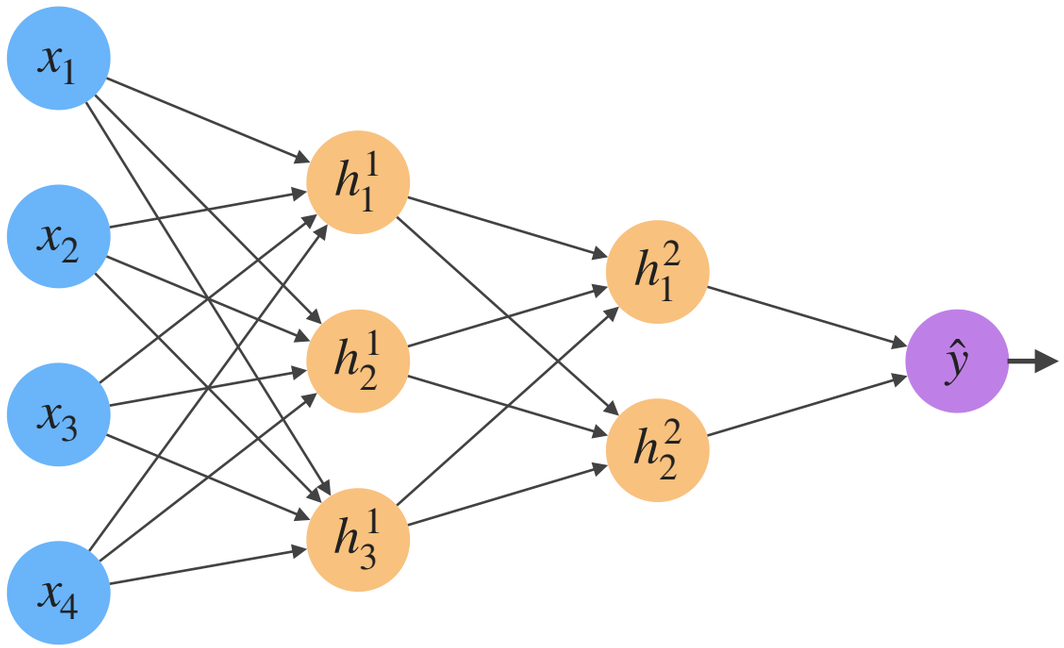
$$h_j^k = f(z_j^k)$$

$$z_j^k = \sum_i w_{i,j}^k \; h_i^{k-1} + b_j$$

Chain rule

$$\frac{\partial E}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; \frac{\partial h_i^k}{\partial z_i^k} \; \frac{\partial z_i^k}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; f'(z_i^k) \; h_j^{k-1}$$

Backpropagation for computing the gradients



$$J(\mathbf{W}) = E$$

$$h_j^k = f(z_j^k)$$

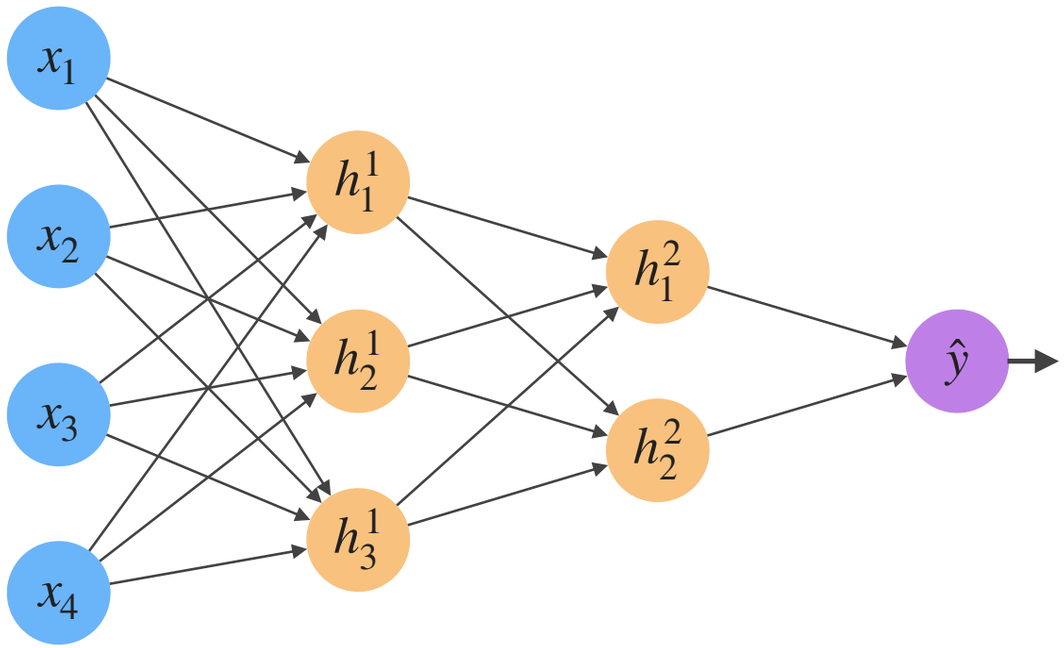
$$z_j^k = \sum_i w_{i,j}^k \; h_i^{k-1} + b_j$$

Chain rule

$$\frac{\partial E}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; \frac{\partial h_i^k}{\partial z_i^k} \; \frac{\partial z_i^k}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; f'(z_i^k) \; h_j^{k-1}$$

$$\frac{\partial E}{\partial h_i^k} = \begin{cases} \frac{\partial E}{\partial \hat{y}}, & \text{if } k \text{ is output layer} \end{cases}$$

Backpropagation for computing the gradients



$$J(\mathbf{W}) = E$$

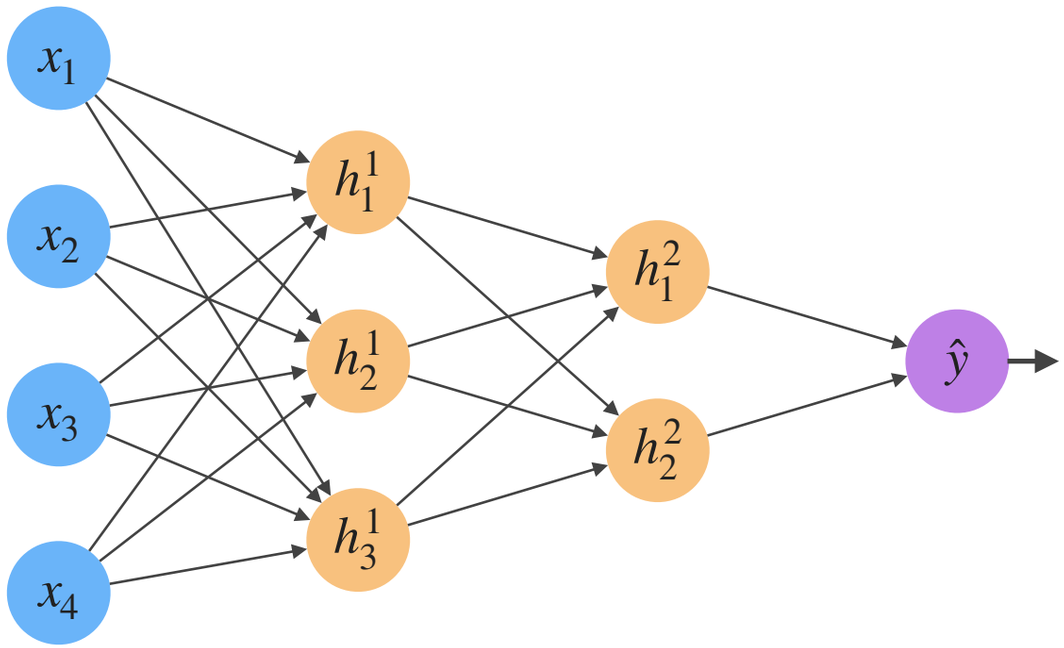
$$h_j^k = f(z_j^k)$$

$$z_j^k = \sum_i w_{i,j}^k \; h_i^{k-1} + b_j$$

Chain rule $$\frac{\partial E}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; \frac{\partial h_i^k}{\partial z_i^k} \; \frac{\partial z_i^k}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; f'(z_i^k) \; h_j^{k-1}$$

$$\frac{\partial E}{\partial h_i^k} = \begin{cases} \dfrac{\partial E}{\partial \hat{y}}, & \text{if } k \text{ is output layer} \\ \displaystyle\sum_{n \in N} \frac{\partial E}{\partial h_n} \; \frac{\partial h_n}{\partial z_n} \; \frac{\partial z_n}{\partial h_i^k}, & \text{otherwise} \end{cases}$$

Backpropagation for computing the gradients



$$J(\mathbf{W}) = E$$

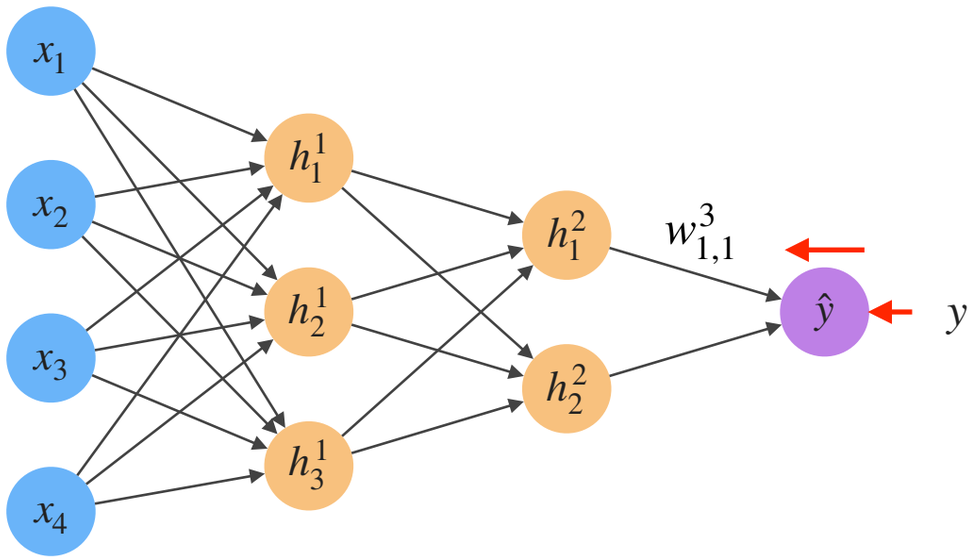
$$h_j^k = f(z_j^k)$$

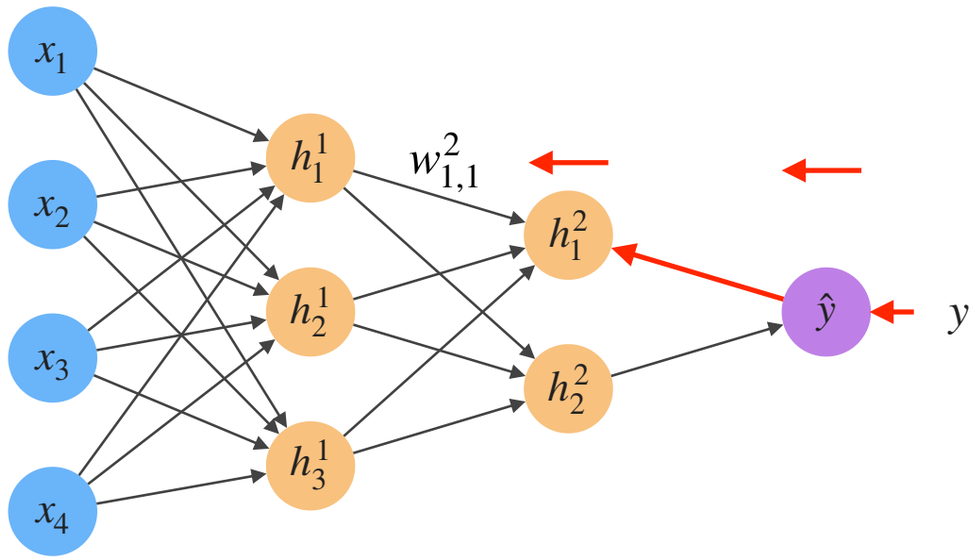
$$z_j^k = \sum_i w_{i,j}^k \; h_i^{k-1} + b_j$$

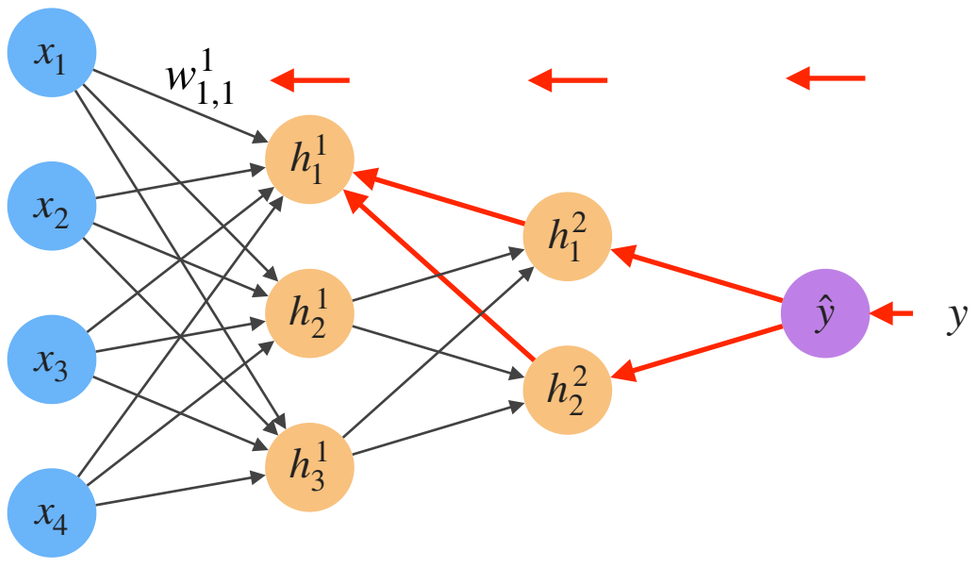
Chain rule $$\frac{\partial E}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; \frac{\partial h_i^k}{\partial z_i^k} \; \frac{\partial z_i^k}{\partial w_{j,i}^k} = \frac{\partial E}{\partial h_i^k} \; f'(z_i^k) \; h_j^{k-1}$$

$$\frac{\partial E}{\partial h_i^k} = \begin{cases} \frac{\partial E}{\partial \hat{y}}, & \text{if } k \text{ is output layer} \\ \sum_{n \in N} \frac{\partial E}{\partial h_n} \; \frac{\partial h_n}{\partial z_n} \; \frac{\partial z_n}{\partial h_i^k} = \sum_{n \in oc\_N} \frac{\partial E}{\partial h_n} \; f'(z_n) \; w_{i,n}^{k+1}, & \text{otherwise} \end{cases}$$

Backpropagation for computing the gradients







Techniques for easier training of Neural Networks

- Vanishing gradients
  - For $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$, we have $f'(x) = \sigma(x)(1-\sigma(x))$
  - When x is too small or too big, the gradient reaches values near zero
  - The weights of lower layers will stop being updated

Techniques for easier training of Neural Networks

- Vanishing gradients
  - For $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$, we have $f'(x) = \sigma(x)(1-\sigma(x))$
  - When x is too small or too big, the gradient reaches values near 0
  - The weights of lower layers will stop being updated

- Saturation
  - The absolute value of the neuron's linear part becomes large
  - The output of the sigmoid is 1 or 0, and the gradient close to 0
  - This would prevent the neuron from learning

Techniques for easier training of Neural Networks

- Vanishing gradients
  - For $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$, we have $f'(x) = \sigma(x)(1-\sigma(x))$
  - When x is too small or too big, the gradient reaches values near 0
  - The weights of lower layers will stop being updated

- Saturation
  - The absolute value of the neuron's linear part becomes large
  - The output of the sigmoid is 1 or 0, and the gradient close to 0
  - This would prevent the neuron from learning

$\rightarrow$ Non-saturating activation functions (ReLU, Leaky ReLU, ...)

Techniques for easier training of Neural Networks

- Overfitting due to large capacity

Techniques for easier training of Neural Networks

- Overfitting due to large capacity

→ Dropout regularization

  - During training, a fraction of the neurons is randomly disabled
  - The activation of the remaining neurons are scaled up

Techniques for easier training of Neural Networks

- Overfitting due to large capacity

→ Dropout regularization

- During training, a fraction of the neurons is randomly disabled
- The activation of the remaining neurons are scaled up

- Advantages:
  - Form of ensemble training
  - Reduces the reliance of neurons on each other
  - Form of regularization

Techniques for easier training of Neural Networks

- Batch Gradient Descent
  - Compute the average gradient over all data points
  - Pass all data points forward and backward before updating weights
  - Infrequent but representative updates

- Stochastic Gradient Descent
  - Compute the gradient for one randomly selected data point
  - Update weights based on this gradient
  - Frequent but noisy updates - Overall: Faster conversion

Techniques for easier training of Neural Networks

- Batch Gradient Descent
  - Compute the average gradient over all data points
  - Pass all data points forward and backward before updating weights
  - Infrequent but representative updates

- Stochastic Gradient Descent
  - Compute the gradient for one randomly selected data point
  - Update weights based on this gradient
  - Frequent but noisy updates - Overall: Faster conversion

→ Mini-batch Gradient Descent

  - In-between solution
  - Select N random data points and do batch training with these points

Techniques for easier training of Neural Networks

- Challenges of Gradient Descent
  - Getting stuck in valleys and around saddle points
    → Using momentum with SGD
  - Sensitive to the choice of the learning rate
    → Adaptive learning rate schedules, e.g., Adagrad, RMSProp, Adadelta

Potential of DL for RS

- Feature extraction
- Modularity
- Sequence processing
- Complex models

Potential of DL for RS

- Feature extraction
- Modularity
- Sequence processing
- Complex models

However,

- Requires a lot of computational power
- Training time is high
- Black box models

Research directions in DL and RS

- Learning item embeddings
- Deep Collaborative Filtering
- Feature extraction from content
- Session-based recommendations

Partly based on the chapter "Cutting-Edge Collaborative Recommendation Algorithms: Deep Learning" by Balázs Hidasi, published in "Collaborative Recommendations: Algorithms, Practical Challenges, and Applications" by Berkovsky et al., 2018.

Learning Item Embeddings

Word2vec

- Shallow model, used for learning word embeddings
- Linear operations in the embedding space associated with semantics
  - *king - man + woman* $\approx$ *queen*
  - *Paris - France + Italy* $\approx$ *Rome*
- Data
  - (*target word*, *context*) pairs
  - Extracted from documents, where *context* is the set of words appearing next to the *target word*
- Models
  - Continuous Bag Of Words (CBOW)
  - Skip-gram

"Efficient Estimation of Word Representations in Vector Space", Mikolov et al., ICLR 2013.

Word2vec - CBOW

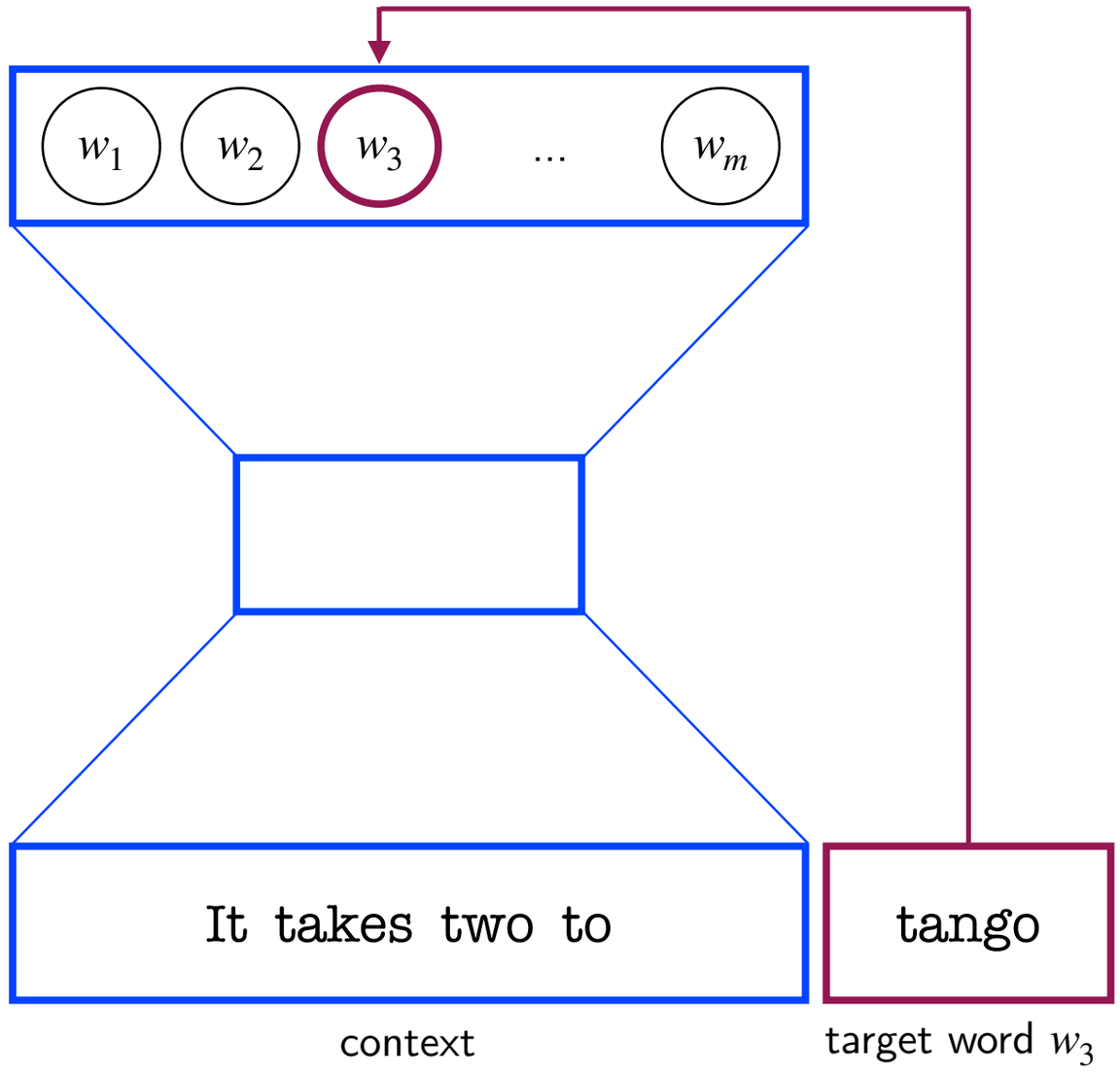
[Output] Softmax classifier

[W_2]

[Hidden] Sum of context's word embeddings

[W_1] Word embedding matrix

[Input] One hot encoding of context



Word2vec - Skip-gram

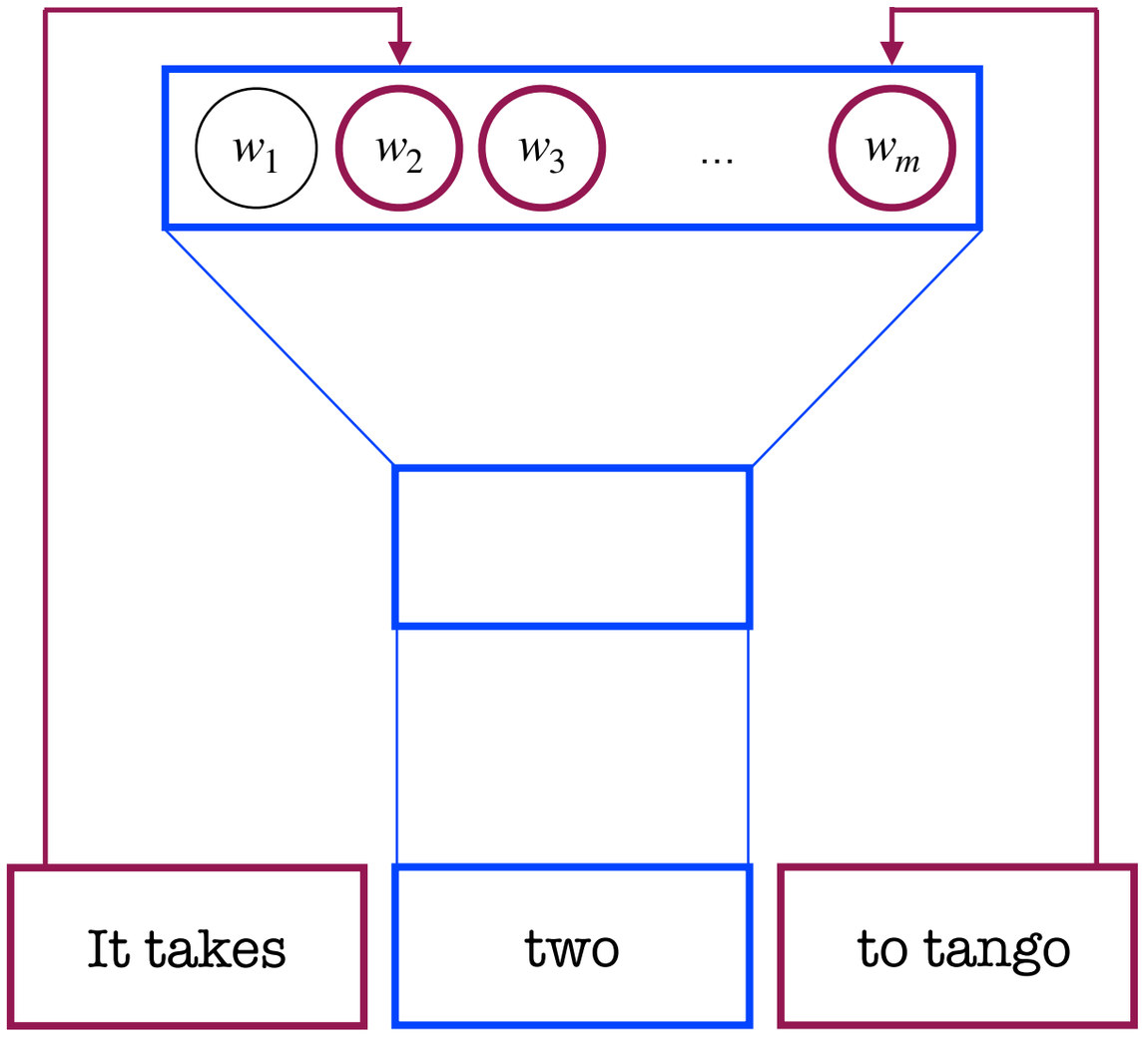
[Output] Softmax classifier

[W_2]

[Hidden] Sum of context's word embeddings

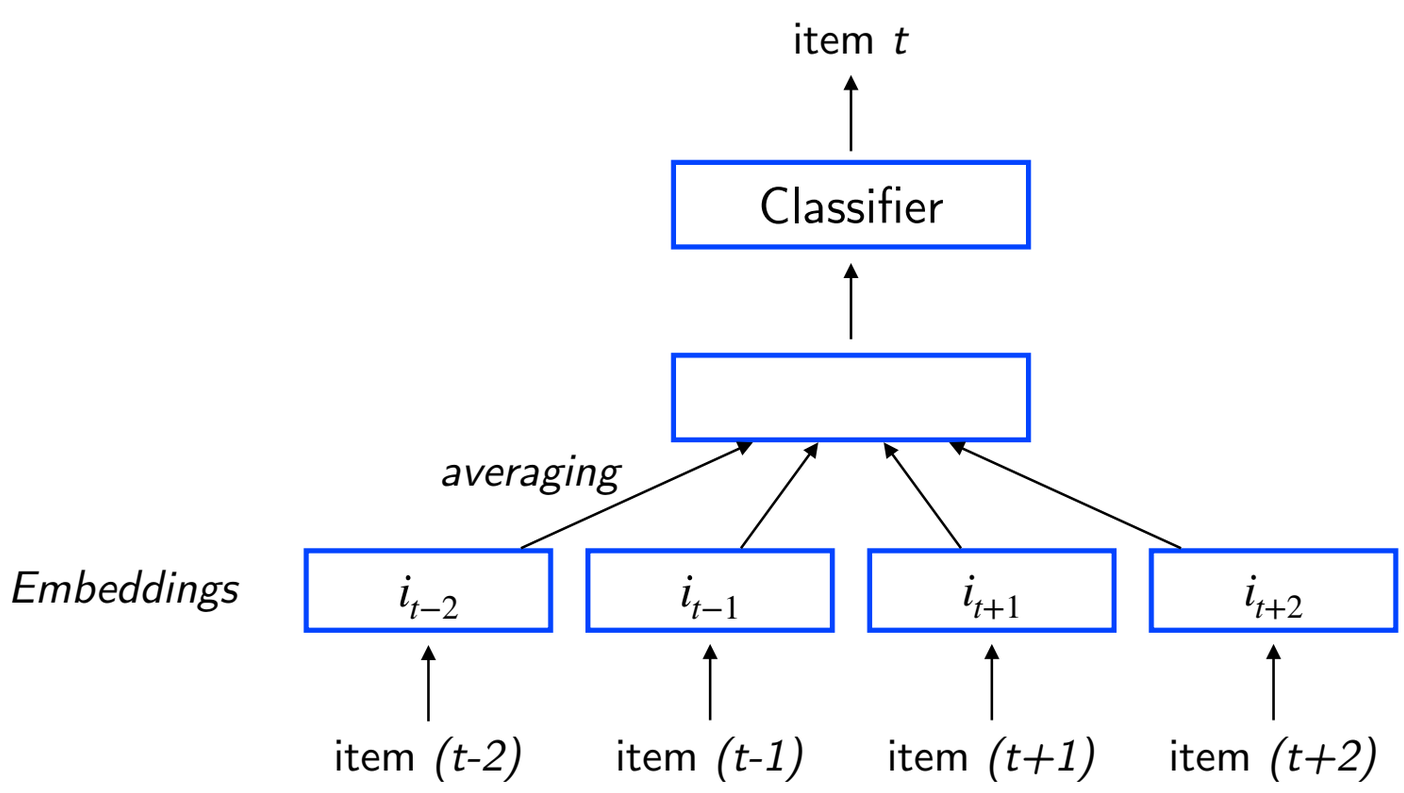
[W_1] Word embedding matrix

[Input] One hot encoding of context



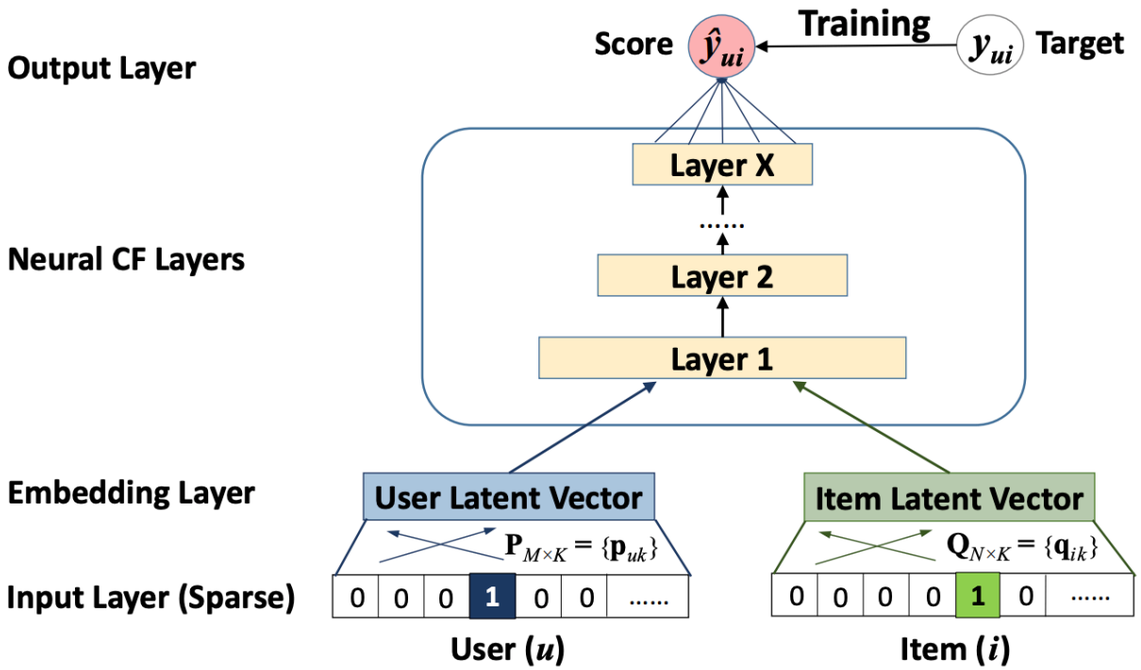
2vec approaches for recommendation

- Replace:
  - Words with items;
  - Documents or contexts with sessions or user profiles



Deep Collaborative Filtering

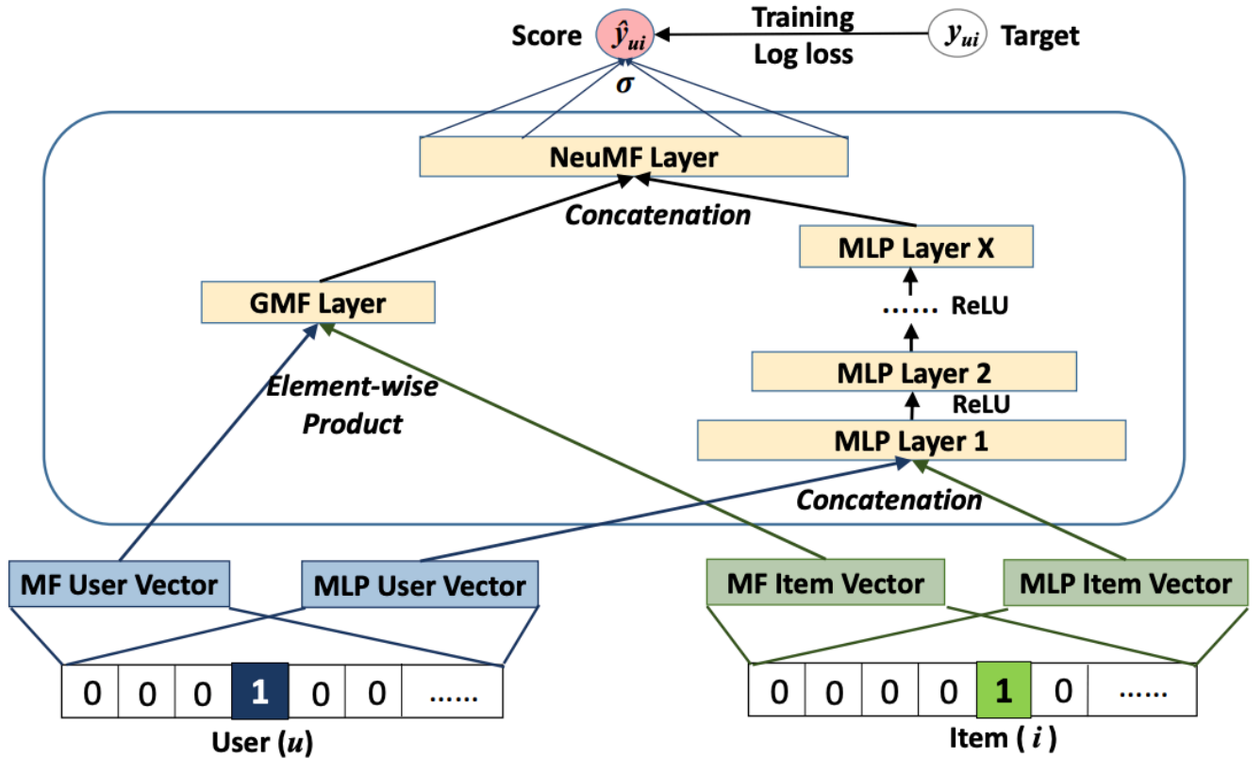
Neural Collaborative Filtering



NCF replaces the dot product used in MF with a neural architecture to capture higher-order interactions

"Neural Collaborative Filtering", He et al., WWW 2017.

Neural Matrix Factorization

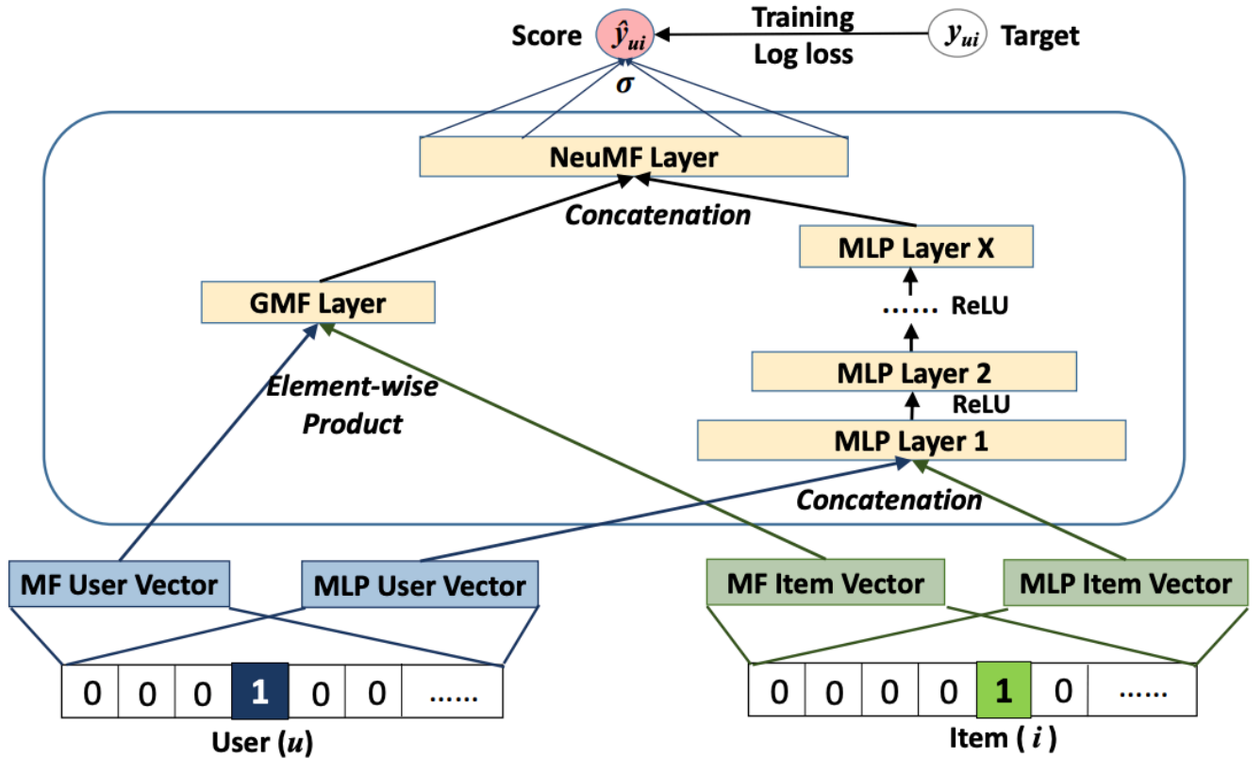


NeuMF combines General Matrix Factorization (GMF) and MLP components:

- GMF Layer: $GMF(p_u, q_i) = p_u \cdot q_i$
- MLP Layers: $MLP(p_u, q_i) = DNN(p_u \| q_i)$
- NeuMF Layer: $\hat{y}_{ui} = \sigma(\mathbf{w}_{GMF} GMF(p_u, q_i) + \mathbf{w}_{MLP} MLP(p_u, q_i) + b)$

"Neural Collaborative Filtering", He et al., WWW 2017.

Neural Matrix Factorization



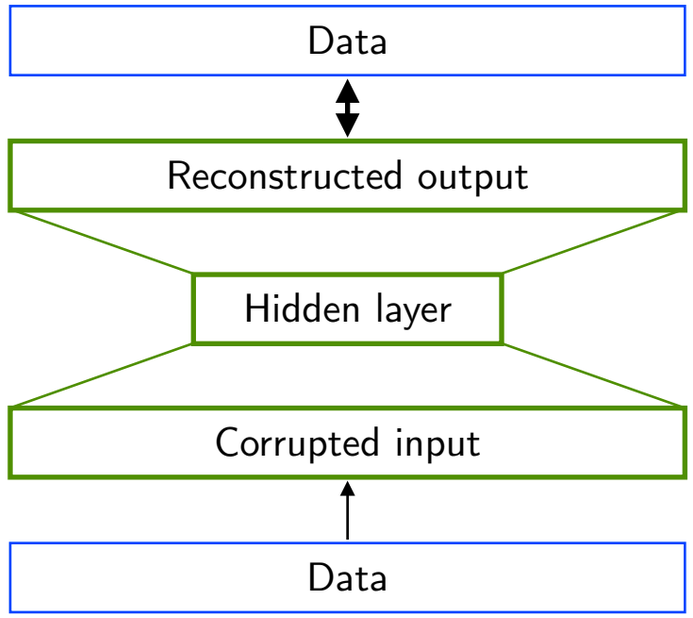
Trained with:

- weighted squared loss for explicit feedback,
- binary cross-entropy loss for implicit feedback

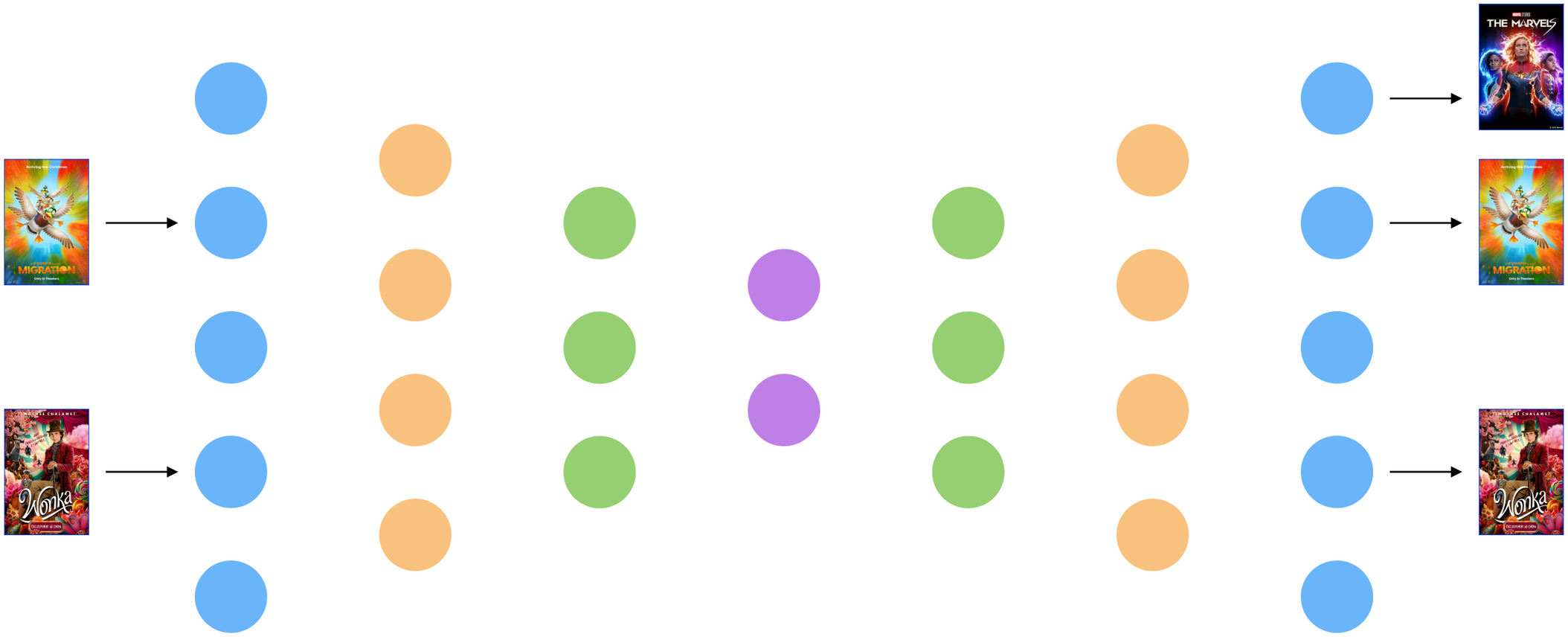
"Neural Collaborative Filtering", He et al., WWW 2017.

Autoencoders

- Autoencoder
  - One hidden layer
  - Same number of input and output units
  - Goal: Reconstruct input on output
  - Hidden layer: compressed representation of data
- Improving generalization
  - Sparse autoencoder
  - Denoising autoencoder
- Deep autoencoders
  - Stacked autoencoders



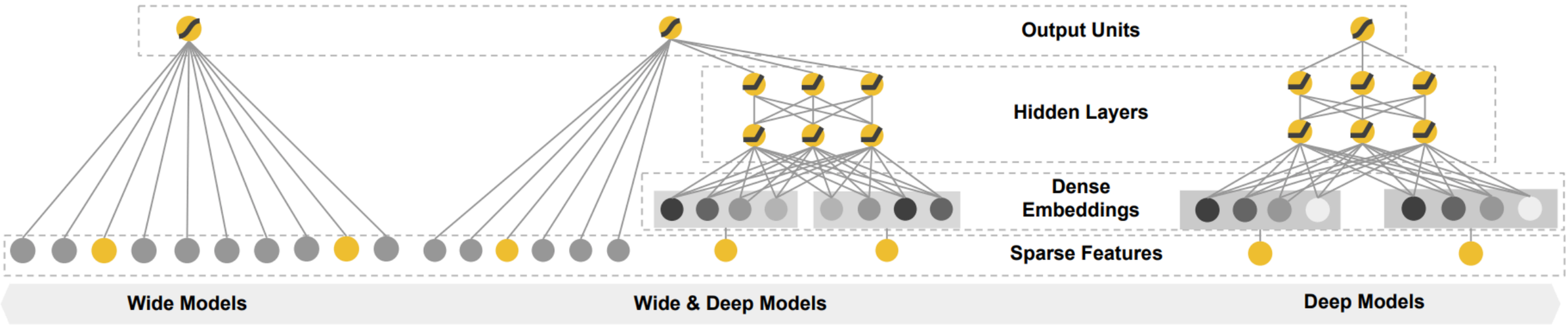
Denoising autoencoders for collaborative filtering



"Collaborative Denoising Auto-Encoders for Top-N Recommender Systems", Wu, Dubois, Zheng, and Ester, WSDM 2016.

Wide & Deep Learning

Google

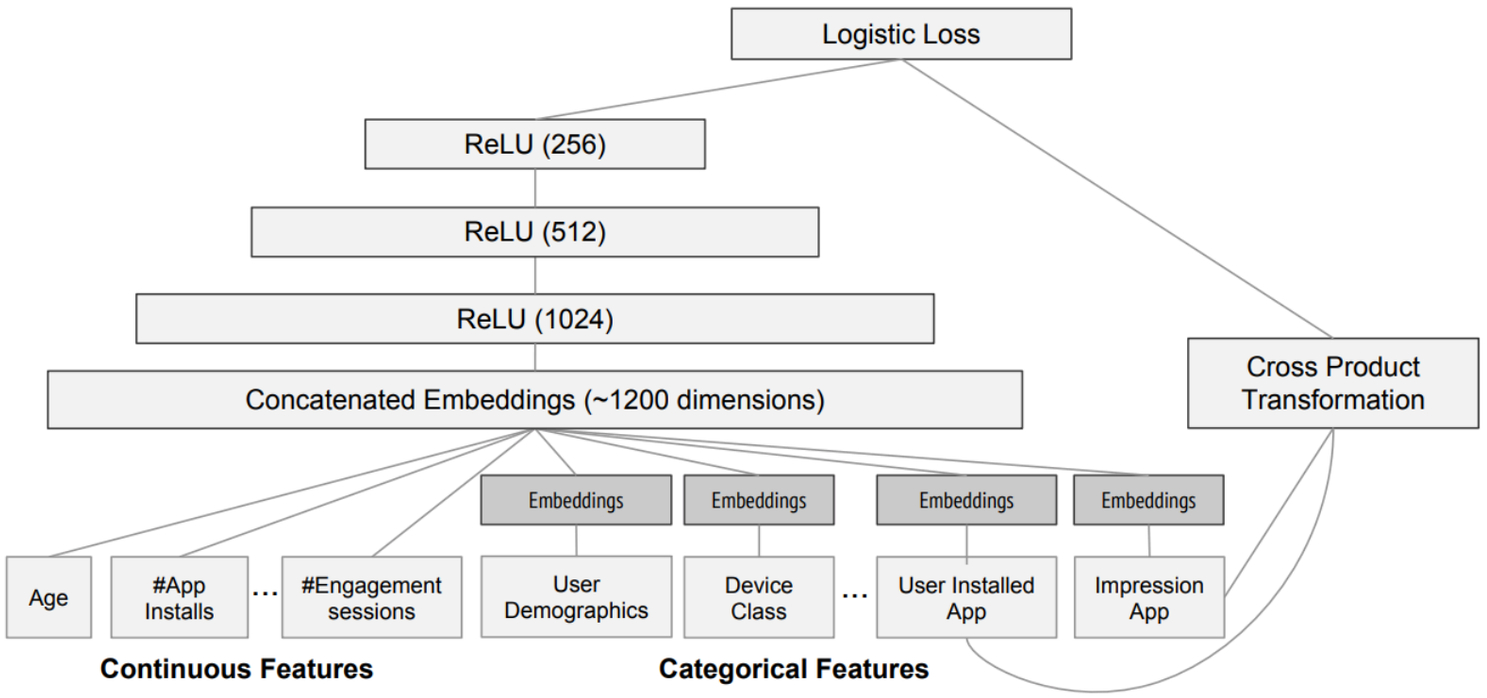


- Wide component, linear model handling memorisation
- Deep component, a deep neural network handling generalisation

Wide & Deep architecture captures the strengths of both

"Wide & Deep Learning for Recommender Systems", Cheng et al., DLRS 2016.

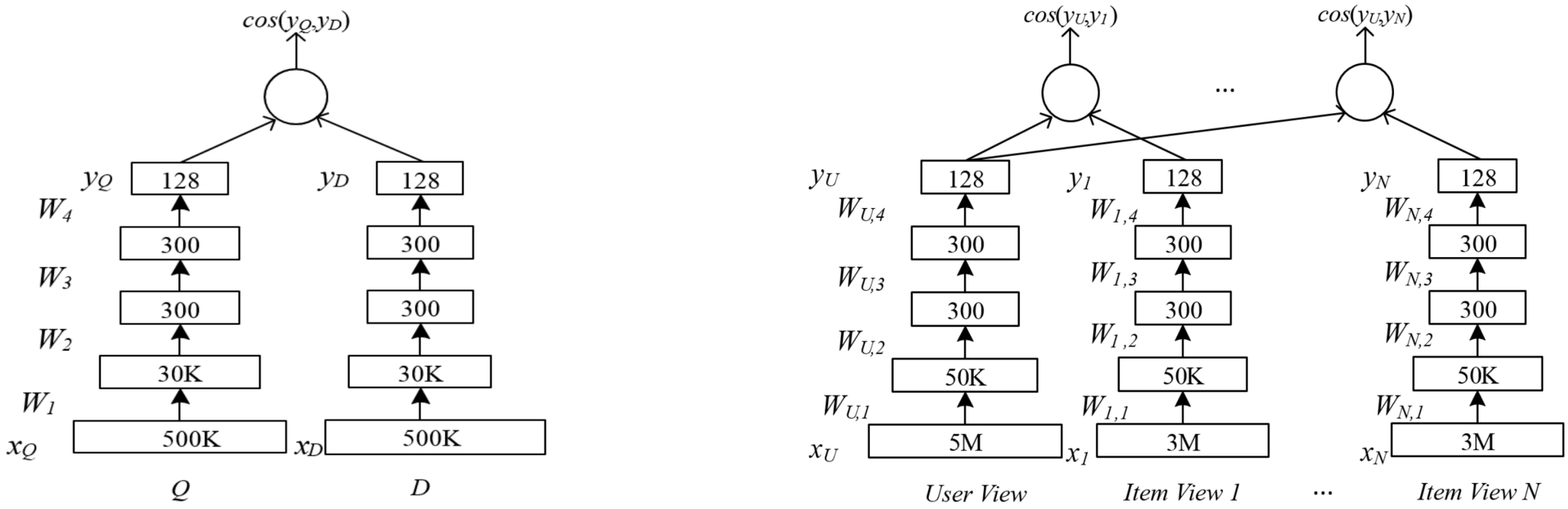
Wide & Deep Learning



Model used for app recommendation for the Google play store

"Wide & Deep Learning for Recommender Systems", Cheng et al., DLRS 2016.

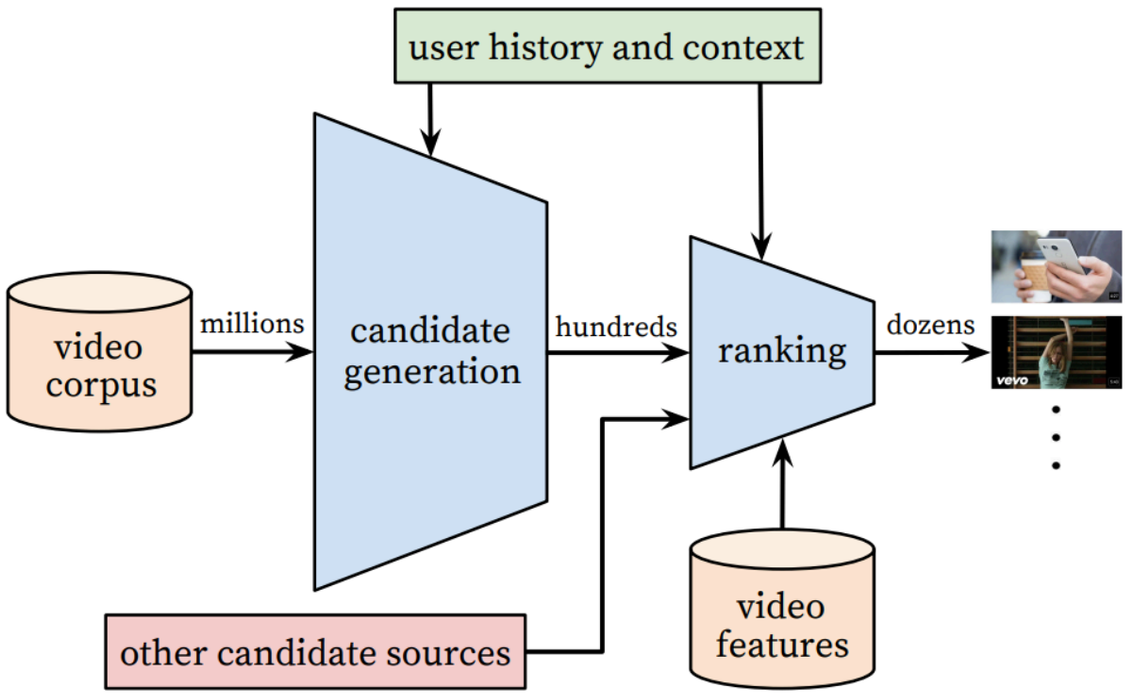
Multi-view deep neural network



Cross-domain user modeling leveraging multiple sources of information or "views" of users

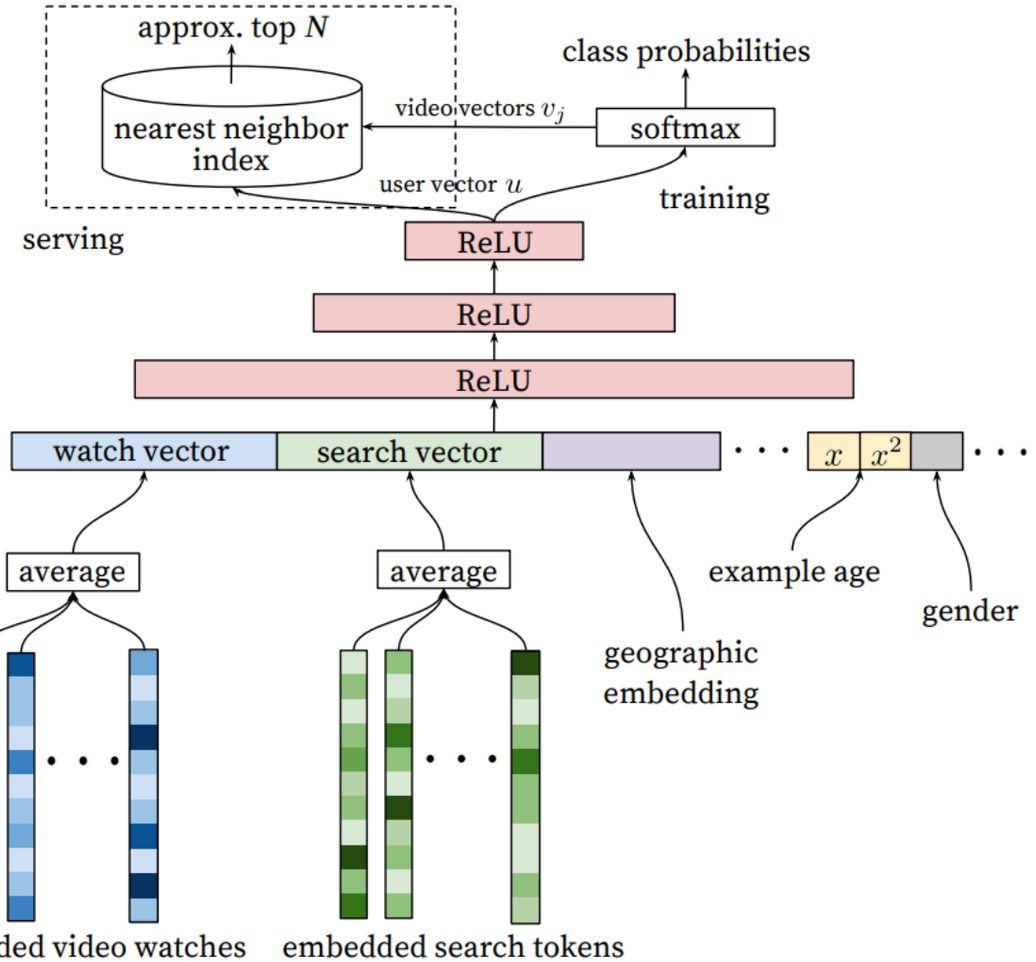
"A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommender Systems", Elkahky, Song, and He, WWW 2015.

Two-Stage Approach for Recommendation



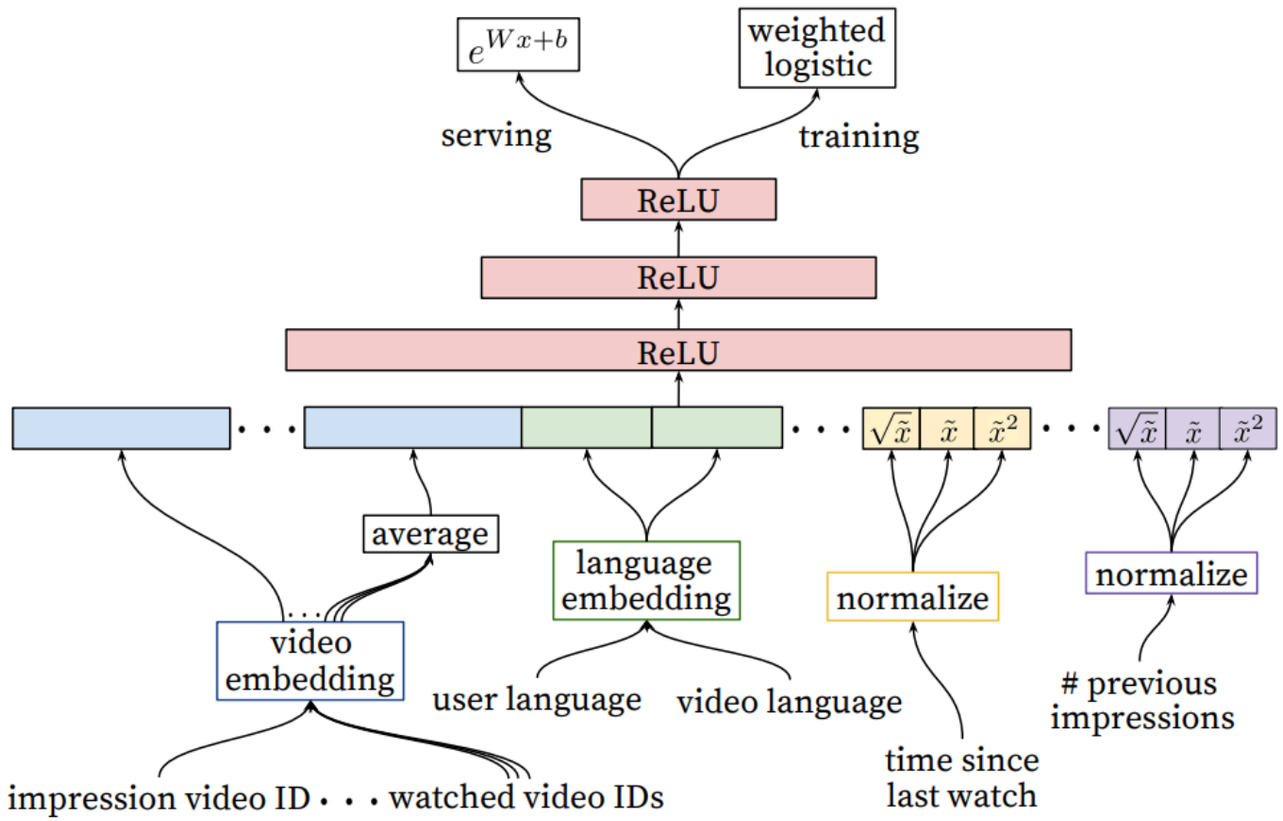
"Deep Neural Networks for YouTube Recommendations", Covington et al., RecSys 2016.

Deep Candidate Generation



"Deep Neural Networks for YouTube Recommendations", Covington et al., RecSys 2016.

Deep Ranking



"Deep Neural Networks for YouTube Recommendations", Covington et al., RecSys 2016.

Feature Extraction from Content

Combining collaborative and content-based filtering

- Weighting
  - Final score as a weighted combination of CF and CBF

Combining collaborative and content-based filtering

- Weighting
  - Final score as a weighted combination of CF and CBF
- Switching
  - Switching between CF and CBF

Combining collaborative and content-based filtering

- Weighting
  - Final score as a weighted combination of CF and CBF
- Switching
  - Switching between CF and CBF
- Intializing
  - Features extracted with CBF used to initialize features in CF

Combining collaborative and content-based filtering

- Weighting
  - Final score as a weighted combination of CF and CBF
- Switching
  - Switching between CF and CBF
- Intializing
  - Features extracted with CBF used to initialize features in CF
- Joint training
  - One feature representation learned while jointly optimizing two problems

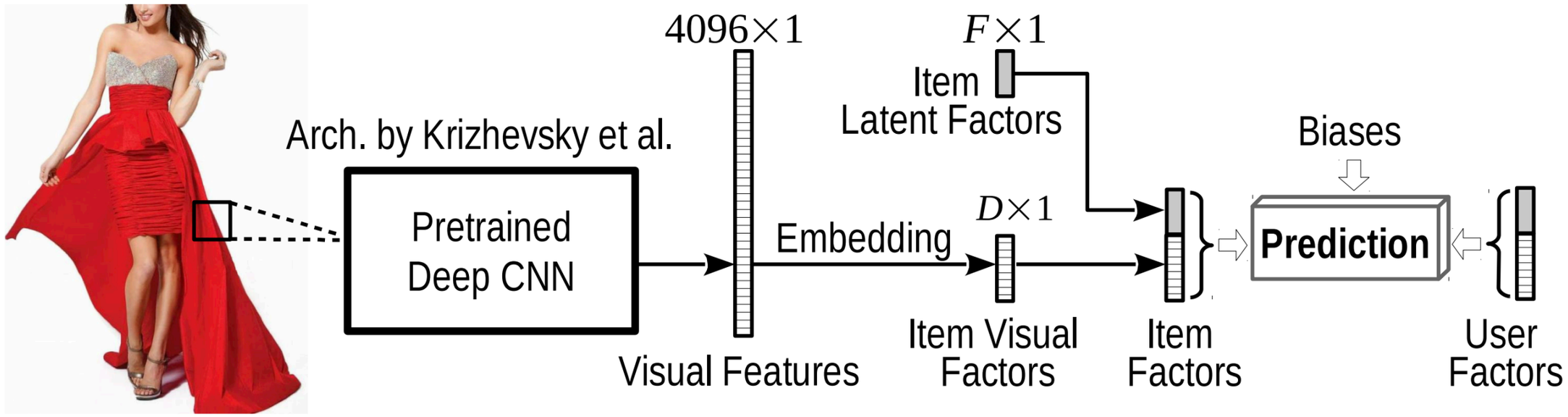
Combining collaborative and content-based filtering

- Weighting
  - Final score as a weighted combination of CF and CBF
- Switching
  - Switching between CF and CBF
- Intializing
  - Features extracted with CBF used to initialize features in CF
- Joint training
  - One feature representation learned while jointly optimizing two problems
- Regularizing
  - Difference between the two representations added to the loss function

Combining collaborative and content-based filtering

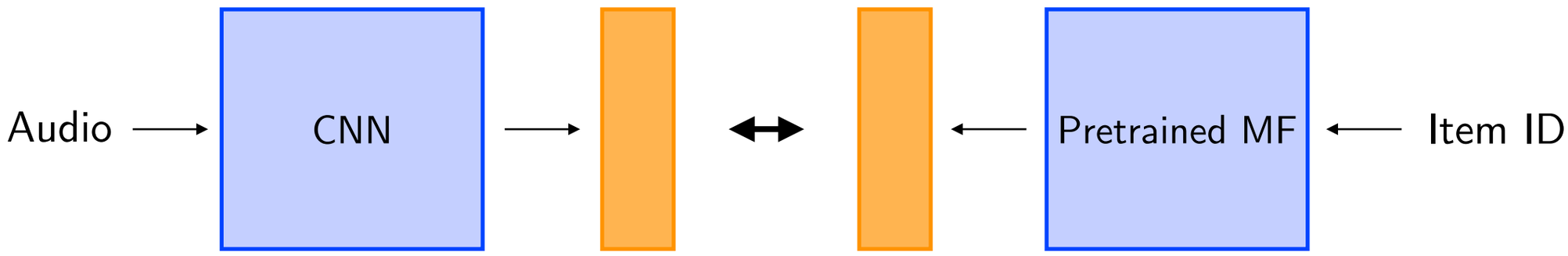
- Weighting
  - Final score as a weighted combination of CF and CBF
- Switching
  - Switching between CF and CBF
- Intializing
  - Features extracted with CBF used to initialize features in CF
- Joint training
  - One feature representation learned while jointly optimizing two problems
- Regularizing
  - Difference between the two representations added to the loss function
- Mapping content
  - Learn mappings from CBF features to CF features

Visual BPR



"VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback", He and Mcauley, AAAI 2016.

Music recommendation based using audio features



- CNN trained given audio spectrograms
- Learning goal of CNN: Item features from MF model
- Applied for new songs to learn mapping between content and CF representations

"Deep Content-Based Music Recommendation", van den Oord et al., NIPS 2013.

Sessions-Based Recommendations

Sessions


De'Longhi 203549 Inissia Cafetière à Capsules Nesp
Marque : De'Longhi
18 735 évaluations | 34 questions avec réponses
Amazon's Choice pour "nespresso machine a café"
Prix : 84,57 €
Tous les prix incluent la TVA.
Payez: 21,62 € x 4 (dont 2.25% de frais inclus) 0% de frais en rejoignar
Disponible à un meilleur prix auprès d'autres vendeurs qui ne propose avec Prime.
Assistance produit Amazon gratuite incluse
Livraison GRATUITE (0,01€ pour les livres) en point retrait. Détails
Neufs & occasions (211) 57,74 € Livraison GRATUITE en France métr
Matériau Plastique
Marque De'Longhi
Passez la souris sur l'image pour zoomer
VIDÉO


Krups YY1500FD Machine à café à Capsules Nesca
Automatique Qualité Professionnelle Pression 15
Chaudes et Froides
Visiter la boutique Krups Dolce Gusto
496 évaluations | 31 questions avec réponses
Amazon's Choice pour "cafetière dolce gusto"
Prix : 109,00 €
Tous les prix incluent la TVA.
Payez: 27,25 € x 4 (-2,45 0 € de frais inclus) Voir détails et conditions
Disponible à un meilleur prix auprès d'autres vendeurs qui ne propo avec Prime.
Assistance produit Amazon gratuite incluse
Passez la souris sur l'image pour zoomer

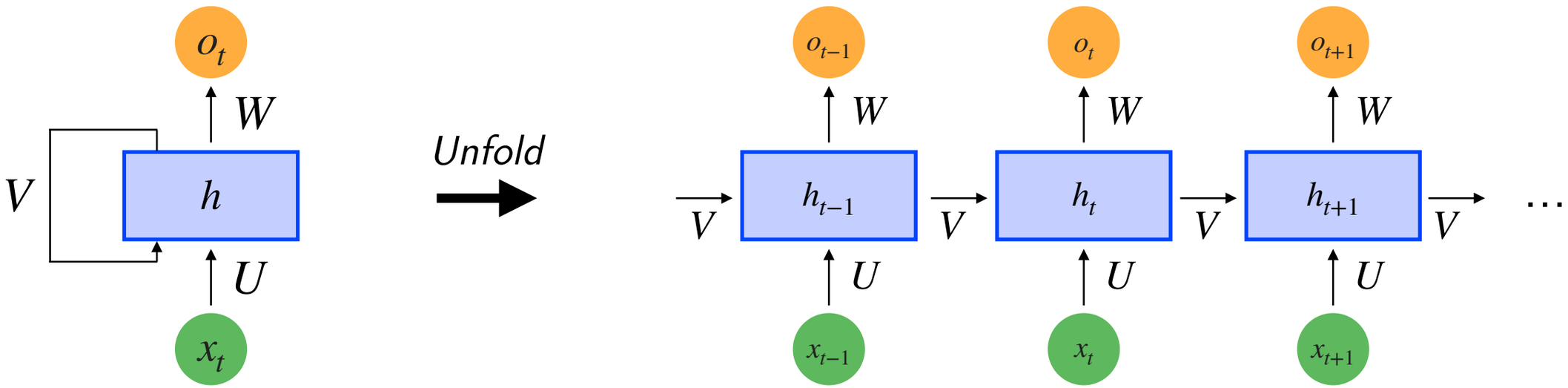

Krups Nescafé Dolce Gusto Piccolo XS rouge, Machine
Cafetière à dosette, Multi-boissons, Intuitive, Pression
KP1A3510
Visiter la boutique Krups Dolce Gusto
13 évaluations
Amazon's Choice pour "dolce gusto machines"
Prix conseillé : 79,99 € De quoi s'agit-il?
Prix : 64,18 €
Économisez : 15,81 € (20 %)
Tous les prix incluent la TVA.
Payez en 4 fois dès 75 € d'achats Voir détails et conditions
Disponible à un meilleur prix auprès d'autres vendeurs qui ne proposent avec Prime.
Passez la souris sur l'image pour zoomer


Melitta Cafetière à Filtre avec Verseuse Isotherme,
d'Arôme, Look Therm Timer, Noir/Acier Brossé, 101
Visiter la boutique Melitta
2 469 évaluations | 55 questions avec réponses
Amazon's Choice pour "cafetiere isotherme"
Ancien prix: 69,00 € De quoi s'agit-il?
Prix : 60,00 €
Économisez : 9,00 € (13 %)
Tous les prix incluent la TVA.
Payez en 4 fois dès 75 € d'achats Voir détails et conditions
Disponible à un meilleur prix auprès d'autres vendeurs qui ne propose avec Prime.
Assistance produit Amazon gratuite incluse
Passez la souris sur l'image pour zoomer
360°


Delonghi ECOV 311.GR Machine à Espresso
Visiter la boutique DeLonghi
524 évaluations | 15 questions avec réponses
Prix conseillé : 179,99 € De quoi s'agit-il?
Prix : 155,92 €
Économisez : 24,07 € (13 %)
Tous les prix incluent la TVA.
Payez: 38,98 € x 4 (3,51 0 € de frais inclus) Voir détails et conditions
Assistance produit Amazon gratuite incluse
Livraison GRATUITE (0,01€ pour les livres) en point retrait. Détails
Style: Unique
Matériau Acier inoxydable
Marque DELONGHI
Passez la souris sur l'image pour zoomer


Bialetti Moka Express noir, en aluminium 1 Tazza Ros
Visiter la boutique Bialetti
10 évaluations
Actuellement indisponible.
Nous ne savons pas quand cet article sera de nouveau approvisionné ni
Couleur: Rose Gold
Taille: 1 Tasse
1 Tasse
3 Tasses
3 tasses
6 Tasses
6 tasses
Style: Unique
Unique
Ensemble Cafetière + Assiette à Induction
Ensemble
Passez la souris sur l'image pour zoomer

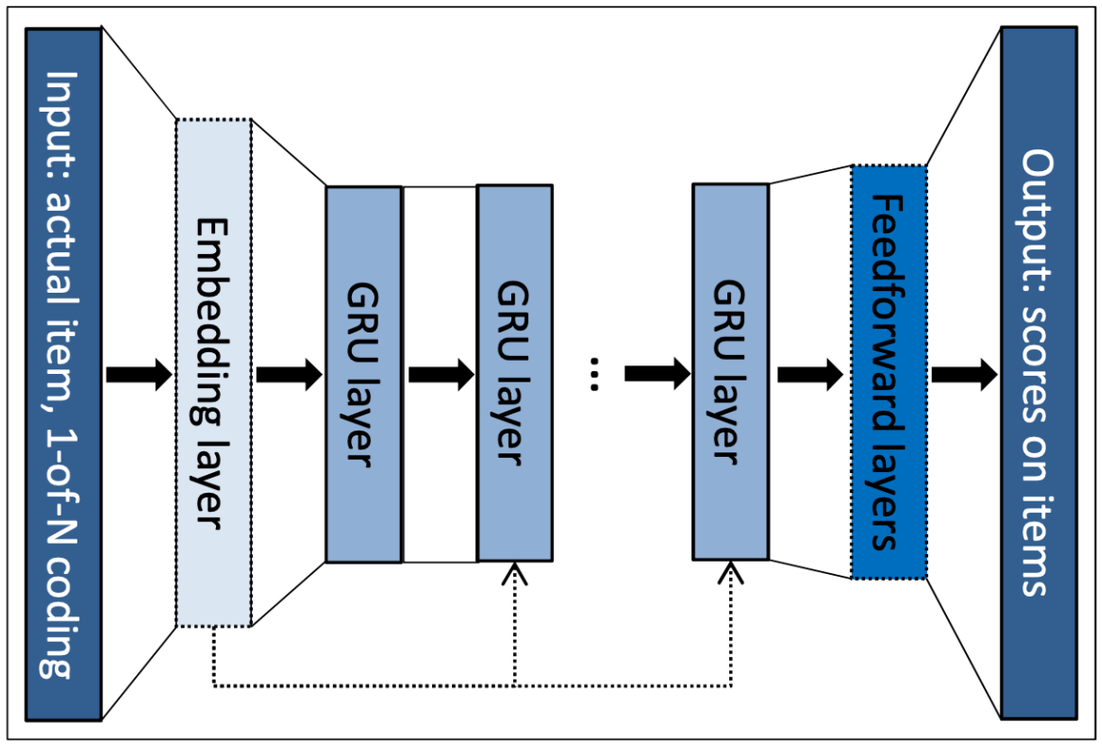
Recurrent Neural Networks



$$s_t = \sigma(Ux_t + Ws_{t-1})$$

$$o_t = softmax(s_t)$$

GRU4Rec



"Session-Based Recommendations with Recurrent Neural Netowrks", Hidasi et al., ICLR 2016.

Conclusion

Research directions in DL and RS

- Learning item embeddings
- Deep Collaborative Filtering
- Feature extraction from content
- Session-based recommendations