Recommender Systems & Embeddings

Alejandro Veloz

Outline

Embeddings

Dropout Regularization

Recommender Systems

Embeddings

Symbolic variable

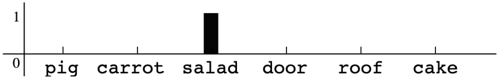
- Text: characters, words, bigrams…
- Recommender Systems: item ids, user ids
- Any categorical descriptor: tags, movie genres, visited URLs, skills on a resume, product categories…

Notation:

Symbol s in vocabulary V

One-hot representation

$$\text{onehot}(\text{'salad'}) = [0, 0, 1, ..., 0] \in \{0, 1\}^{|V|}$$



- Sparse, discrete, large dimension $|V|$
- Each axis has a meaning
- Symbols are equidistant from each other:

euclidean distance = $\sqrt{2}$

Embedding

$$\text{embedding}(\text{'salad'}) = [3.28, -0.45, ...7.11] \in \mathbb{R}^d$$

- Continuous and dense.
- Can represent a huge vocabulary in low dimension, typically: $d \in \{16, 32, ..., 4096\}$.
- Axis have no meaning *a priori*.
- Embedding metric can capture semantic distance.

Neural Networks compute transformations on continuous vectors

Implementation with Keras

Size of vocabulary $n = |V|$, size of embedding d

```
# input: batch of integers
Embedding(output_dim=d, input_dim=n, input_length=1)
# output: batch of float vectors
```

- Equivalent to one-hot encoding multiplied by a weight matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$:

$$\text{embedding}(x) = \text{onehot}(x).\mathbf{W}$$

- $\mathbf{W}$ is typically **randomly initialized**, then **tuned by backprop**
- $\mathbf{W}$ are trainable parameters of the model

Distance and similarity in Embedding space

Euclidean distance

$$d(x, y) = ||x - y||_2$$

- Simple with good properties
- Dependent on norm (embeddings usually unconstrained)

Cosine similarity

$$\text{cosine}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

- Angle between points, regardless of norm
- cosine$(x, y) \in (-1, 1)$
- Expected cosine similarity of random pairs of vectors is 0

Distance and similarity in Embedding space

If x and y both have unit norms:

$$||x - y||_2^2 = 2 \cdot (1 - \text{cosine}(x, y))$$

or alternatively:

$$\text{cosine}(x, y) = 1 - \frac{||x - y||_2^2}{2}$$

Alternatively, dot product (unnormalized) is used in practice as a pseudo similarity

Visualizing Embeddings

- Visualizing requires a projection in 2 or 3 dimensions
- Objective: visualize which embedded symbols are similar

PCA

- Limited by linear projection, embeddings usually have complex high dimensional structure

t-SNE

Visualizing data using t-SNE, L van der Maaten, G Hinton, *The Journal of Machine Learning Research*, 2008

t-Distributed Stochastic Neighbor Embedding

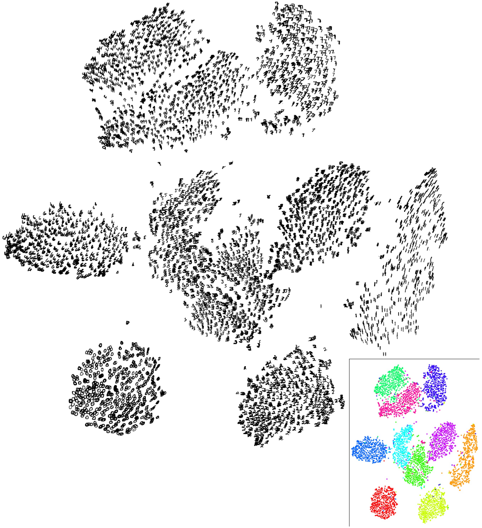
- Unsupervised, low-dimension, non-linear projection
- Optimized to preserve relative distances between nearest neighbors
- Global layout is not necessarily meaningful

t-SNE projection is non deterministic (depends on initialization)

- Critical parameter: perplexity, usually set to 20, 30
- See http://distill.pub/2016/misread-tsne/

Example word vectors

Visualizing Mnist



Dropout Regularization

Regularization

Size of the embeddings

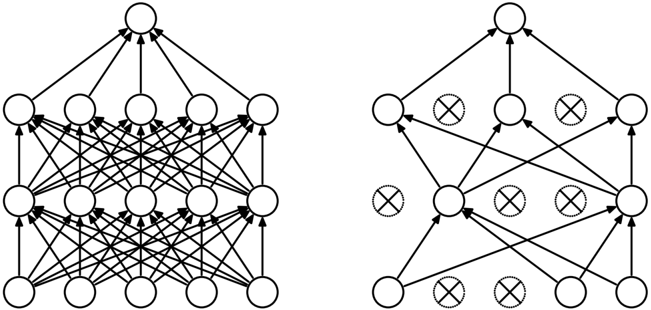
Depth of the network

L_2 **penalty on embeddings**

Dropout

- Randomly set activations to 0 with probability p
- Bernoulli mask sampled for a forward pass / backward pass pair
- Typically only enabled at training time

Dropout



(a) Standard Neural Net (b) After applying dropout.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Srivastava et al., *Journal of Machine Learning Research* 2014

Dropout

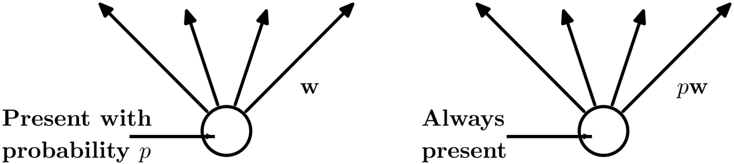
Interpretation

- Reduces the network dependency to individual neurons
- More redundant representation of data

Ensemble interpretation

- Equivalent to training a large ensemble of shared-parameters, binary-masked models
- Each model is only trained on a single data point

Dropout

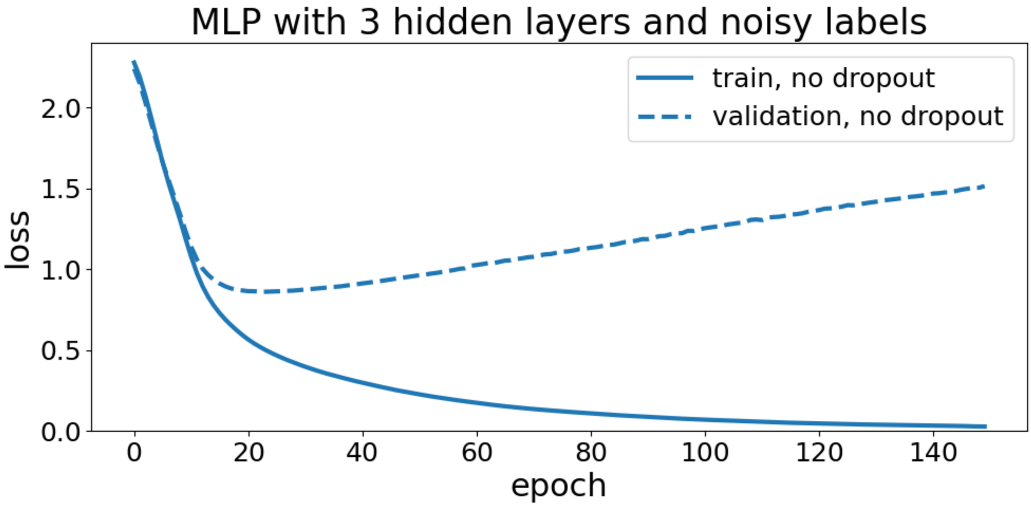


(a) At training time (b) At test time

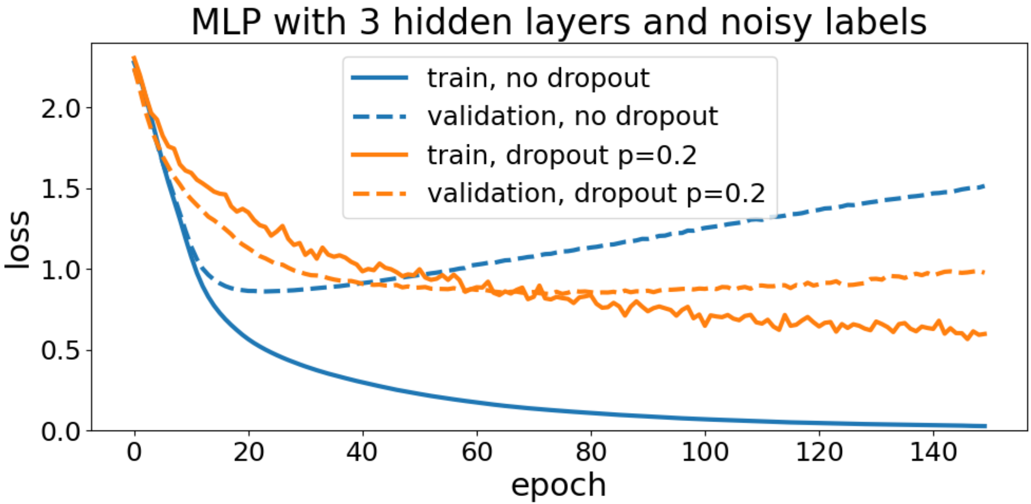
At test time, multiply weights by p to keep same level of activation.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Srivastava et al., *Journal of Machine Learning Research* 2014

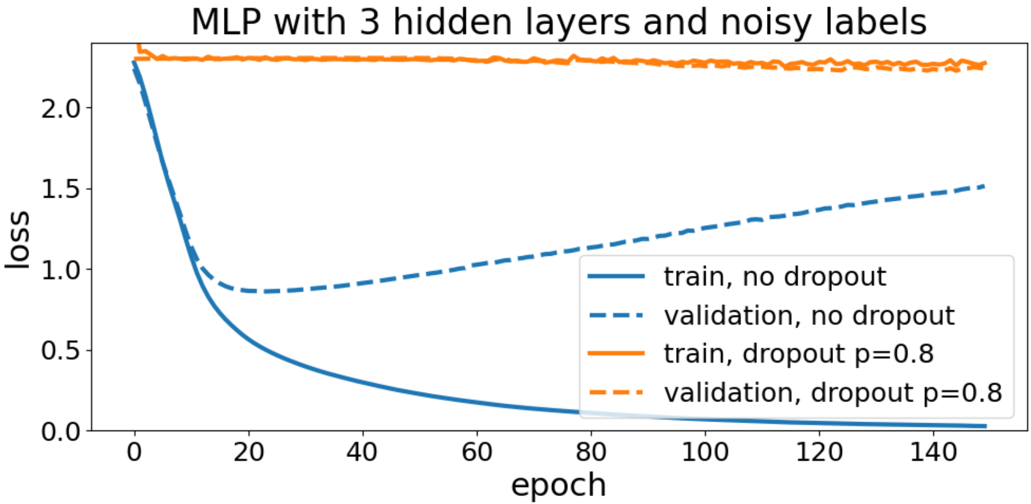
Overfitting Noise



A bit of Dropout



Too much: Underfitting



Implementation with Keras

```
model = Sequential()
model.add(Dense(hidden_size, input_shape, activation='relu'))
model.add(Dropout(p=0.5)) # <=========
model.add(Dense(hidden_size, activation='relu'))
model.add(Dropout(p=0.5)) # <=========
model.add(Dense(output_size, activation='softmax'))
```

Recommender Systems

Recommender Systems

Recommend contents and products

Movies on Netflix and YouTube, weekly playlist and related Artists on Spotify, books on Amazon, related apps on app stores, “Who to Follow” on twitter…

Prioritized social media status updates

Personalized search engine results

Personalized ads and RTB

RecSys

Content-based vs Collaborative Filtering (CF)

Content-based: user metadata (gender, age, location…) and item metadata (year, genre, director, actors)

Collaborative Filtering: passed user/item interactions: stars, plays, likes, clicks

Hybrid systems: CF + metadata to mitigate the cold-start problem

Explicit vs Implicit Feedback

Explicit: positive and negative feedback

- Examples: review stars and votes
- Regression metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE)…

Implicit: positive feedback only

- Examples: page views, plays, comments…
- Ranking metrics: ROC AUC, precision at rank, NDCG…

Explicit vs Implicit Feedback

Implicit feedback much more **abundant** than explicit feedback

Explicit feedback does not always reflect **actual user behaviors**

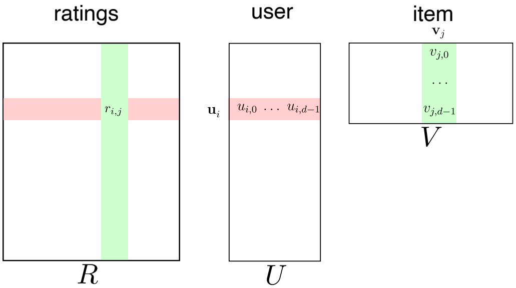
- Self-declared independent movie enthusiast but watch a majority of blockblusters

Implicit feedback can be **negative**

- Page view with very short dwell time
- Click on “next” button

Implicit (and Explicit) feedback distribution **impacted by UI/UX changes** and the **RecSys deployment** itself.

Matrix Factorization for CF

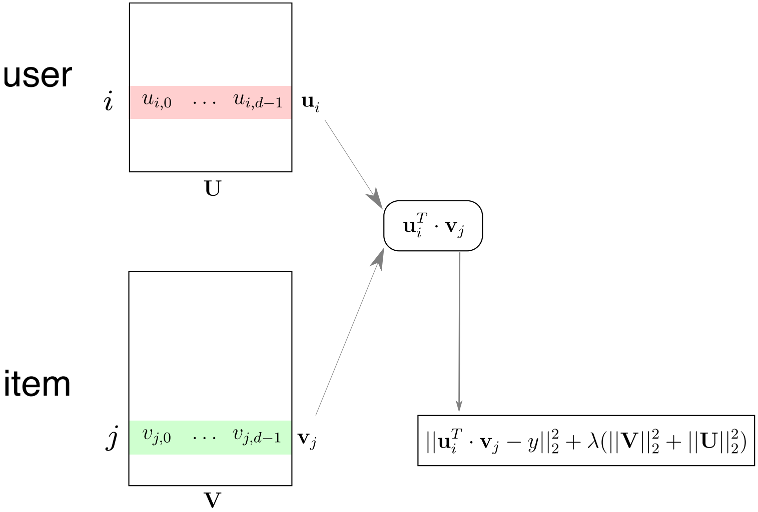


$$L(U,V) = \sum_{(i,j)\in D} \|r_{i,j} - \mathbf{u}_i^\top \cdot \mathbf{v}_j\|_2^2 + \lambda(\|U\|_2^2 + \|V\|_2^2)$$

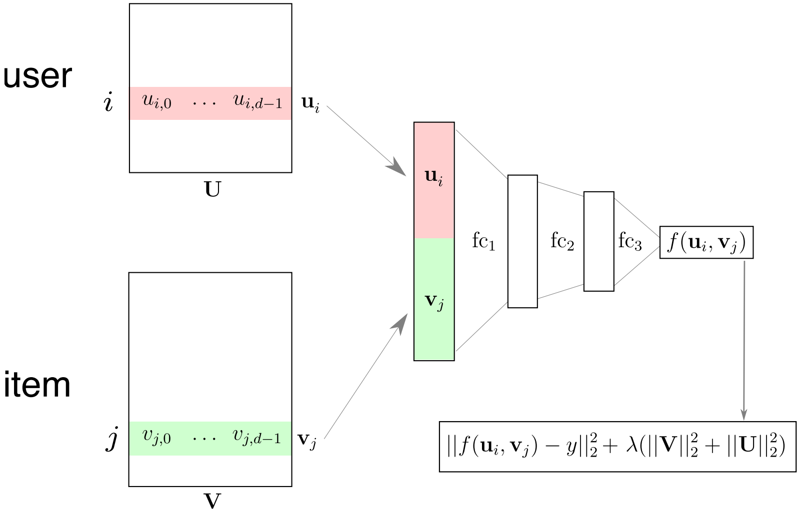
- Train U and V on observed ratings data $r_{i,j}$
- Use $U^\top V$ to find missing entries in sparse rating data matrix R

Architecture and Regularization

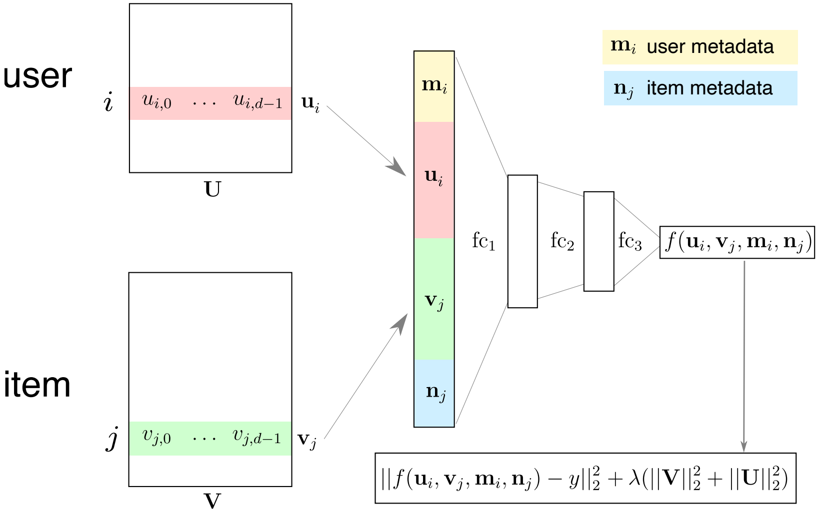
RecSys with Explicit Feedback



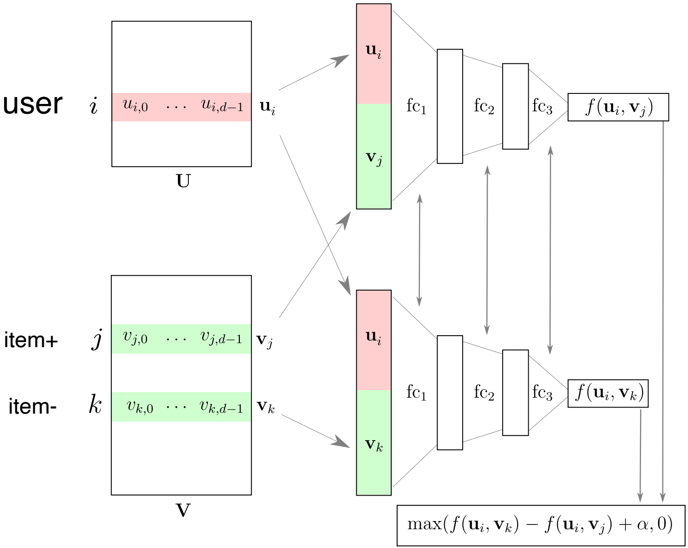
Deep RecSys Architecture



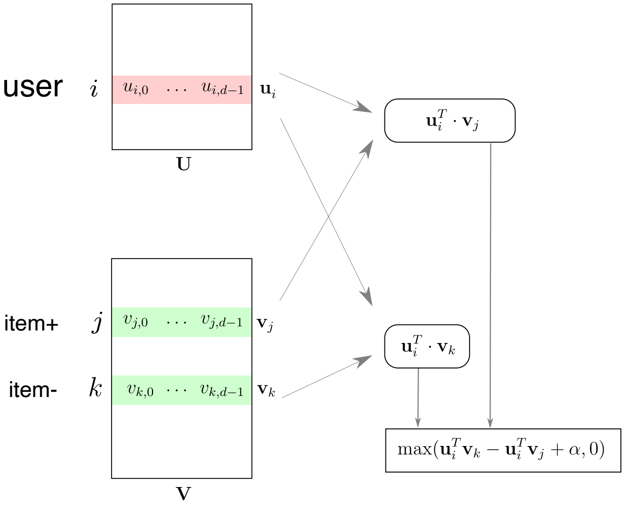
Deep RecSys with metadata



Implicit Feedback: Triplet loss

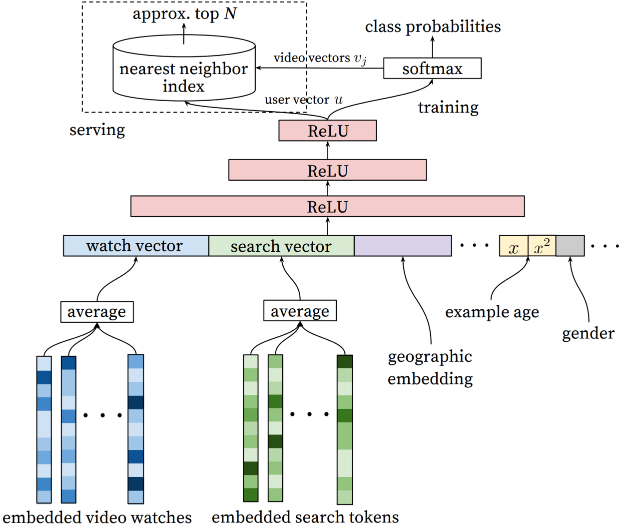


Deep Triplet Networks



Training a Triplet Model

- Gather a set of positive pairs user i and item j
- While model has not converged:
    - Shuffle the set of pairs (i, j)
    - For each (i, j):
        - Sample item k uniformly at random
        - Call item k a negative item for user i
        - Train model on triplet (i, j, k)



Deep Neural Networks for YouTube Recommendations
https://research.google.com/pubs/pub45530.html

Ethical Considerations of Recommender Systems

Ethical Considerations

Amplification of existing discriminatory and unfair behaviors / bias

- Example: gender bias in ad clicks (fashion / jobs)
- Using the firstname as a predictive feature

Amplification of the filter bubble and opinion polarization

- Personalization can amplify “people only follow people they agree with”
- Optimizing for “engagement” promotes content that causes strong emotional reaction (and turns normal users into *haters*?)
- RecSys can exploit weaknesses of some users, lead to addiction
- Addicted users clicks over-represented in future training data

Call to action

Designing Ethical Recommender Systems

- Wise modeling choices (e.g. use of “firstname” as feature)
- Conduct internal audits to detect fairness issues: SHAP, Integrated Gradients, fairlearn.org
- Learning representations that enforce fairness?

Transparency

- Educate decision makers and the general public
- How to allow users to assess fairness by themselves?
- How to allow for independent audits while respecting the privacy of users?