Embeddings

Embedding
- is a mapping from a discrete objects to vectors of real numbers

Example:
300-dimensional embedding for english words could include

blue: (0.01359, 0.00075997, 0.24608, ..., -0.2524, 1.0048, 0.06259)
blues: (0.01396, 0.11887, -0.48963, ..., 0.033483, -0.10007, 0.1158)
orange: (-0.24776, -0.12359, 0.20986, ..., 0.079717, 0.23865, -0.014213)
oranges: (-0.35609, 0.21854, 0.080944, ..., -0.35413, 0.38511, -0.070976)

Dimensions & Vectors
- individual dimensions have no inherent meaning
- instead: ML algorithm make use of the overall patterns of
  - location
  - distance between vectors

As Input and Output of ML
- Input:
  - ML works on vectors of real numbers
  - train best on dense vectors, in which all values contribute to define an object
  - No natural vector representation: eg text
  - Improved representation: categorial vars => transforms discrete objs into continuous vectors
- Output:
  - embeddings map objects to vectors
  - use similarity in vector space
    eg. Euclidean distance, or angle between vectors as measure of object similarity
  - used to find nearest neighbors
    Three nearest neighbors for each word and the corresponding angles

    blue: (red, 47.6°), (yellow, 51.9°), (purple, 52.4°)
    blues: (jazz, 53.3°), (folk, 59.1°), (bluegrass, 60.6°)
    orange: (yellow, 53.5°), (colored, 58.0°), (bright, 59.9°)
    oranges: (apples, 45.3°), (lemons, 48.3°), (mangoes, 50.4°)

Projections
- t-SNE
  - nonlinear nondeterministic algorithm
  - tries to preserve local neighborhoods in the data
  - at expense of distorting global structure
- PCA:
  - linear deterministic algorithm
  - tries to capture as much of the data variability in as few dimensions as possible.
  - highlights large-scale structure in the data, but can
  - at expense of distorting local neighborhoods

Encoding categorial variables:
From Integer & One-Hot Encoding to Embeddings

Goal during encoding of categorial variable
Find
1. Necessary: numerical representation in which
2. Optional: semantically similar items (movies or words) have similar distances in vector space

Integer Encoding
- two types of discrete variables: ordinal and nominal
- ordinal: ordered - eg low, medium, high
- nominal: not ordered
- Integer Encoding: often integers are used for convenience to label different states
  => have no information in themselves
- Example: 1, 2, 3 = red, blue, yellow
  - but cannot not assume:
    - blue > red
    - average of red and yellow are blue

Problem: Integer Encoding
- Problem:
  - often no obvious continuity in categorial features of structured data
    => continuous nature of neural networks limits applicability to categorical variables
- Naive approach 1: integer representation
  - use integer representation of category variables
    => does not work well
- Naive approach 2: One-Hot encoding
  - tries to circumvent problem of

One-hot encoding
- used to represent categorial variables or strings
- sparse vector in which:
  - one element = 1
  - other elements = 0

Problem: One-Hot-Encoding (in NN)
1. Size of Network:
   - huge input vectors = very huge number of weights for a neural network
   - M words in vocabulary, N nodes in first layer above the input
     => MxN weights to train for that layer
   - number inputs to network: cardinality of variable - 1
     => example: 1M movies => 1M inputs
   - Problem: large number of weights:
     - Amount of data: more weights => more data needed
     - Amount of computation: more weights => more computation required to train

2. Lack of Meaningful Relations Between Vectors
- relationships of values of the cat variable / strings is not expressed in encoding
- eg. does not encode inherent / intrinsic properties and relationships between the values categorial variables
- treats different values of categorical variables completely independent of each other
  => ignores informative relations between them

==> Solution: Embeddings

Solution: Entity Embedding
- translate large sparse vectors into lower-dim space that preserves semantic relationships
- automatically learns representation of categorical features in multi-dimensional spaces
- puts values with similar effect in ML-function approximation problem close to each other
  => reveals intrinsic continuity & helps NN and ML algorithms

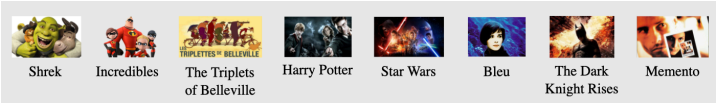
Advantage Embeddings: vs one-hot encoding
- maps similar values close to each other in the embedding space
  => reveals intrinsic properties of the categorical variable
- generalizes better when data is sparse and statistics unknown
  => useful for high cardinality features - other methods tend to overfit
- generated embeddings improves performance of ML methods
  if when used as the input features instead
- entity embedding define distance measure for categorical variables
- reduces memory usage
- speeds up neural networks

Embedding space: Intuition

Embedding: Intuition
- Goal:
  To recommend movies it needs to be determined which movies are similar to each other
- Solution: embedding movies into a low-dimensional space so that similar movies are nearby

Creating embedding by hand

- 1-dimensional embedding
  Arrange movies by „geared towards children versus adults“



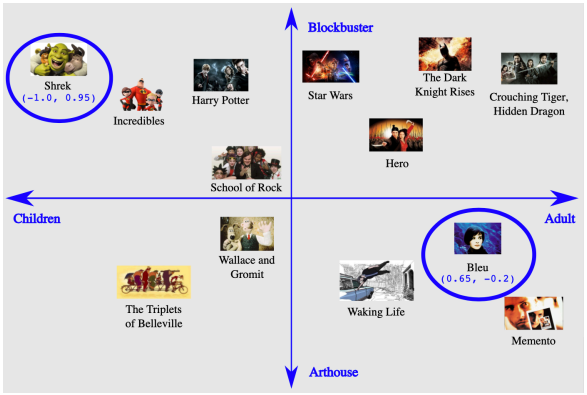
- 2-dimensional embedding
  1. Arrange movies by „geared towards children versus adults“
  2. Arrange movies by „extent to blockbuster movies versus arthouse movies“

  - Achieves:
    - mapping movies into an embedding space
    - where movie is described by a two-dimensional set of coordinates
    - eg. "Shrek" => (-1.0, 0.95) ; "Bleu" => (0.65, -0.2).

  - Similarity
    - two-dimensional embedding defines a distance between movies
    - movies are nearby => inferred to be similar

  - Latent dimension
    - each dimension represents a feature
      that is not explicit in the data but rather inferred from it
    - values along a dimension are not meaningful by itself
      BUT distances between movies in embedding space is important

=> Latent dimension are **usually not found by hand / intuition**
but can be automatically **computed by neural networks or matrix factorization**



NN as feature learners for categorical variables
- EE using NN can be used to learn feature representations for categorical variables
- which are commonly observed in structured data
- high-cardinality variables benefit the most

NN and structured data
- NN mostly used on unstructured data - eg. vision, speech, nlp
- BUT snot often used on structured data
- on Kaggle: tree-based methods beat neural networks

=> reason - type of function learned

Types of functions learned
- ML learns: $y = f(x_1, x_2, \ldots, x_n)$.
- Neural network:
  - can approximate any continuous function and piecewise continuous function
  - not suitable to approximate arbitrary non-continuous functions
    => assumes certain level of continuity in its general form
  - continuity of data:
    - during optimization: guarantees the convergence
    - during prediction: changing values of input keeps output stable
- Decision trees:
  - not assume any continuity of feature variables
    => divide states of a variable as fine as necessary
- Problems in nature:
  - often continuous, IF appropriate representation is found

Reveal continuity by other representation
- Problems in nature often continuous ONLY IF good representation is found
  => if better way to reveal continuity of data found => increases the power of NN
- eg. CNN
  - group pixels in the same neighborhood together
  - increases continuity of data
    vs simply representing image as a flattened vector of pixel values
- eg. word embeddings NLP:
  - puts words with similar meaning closer to each other in a word space
  - increases the continuity of the words compared to one-hot encoding of words

Sparse and high-cardinality data
- embeddings used for sparse feature matrices and categorial variables with high card.
- Sparse: only few entries in table / column - eg. recommender system
- high-cardinality: many factors levels - eg. city

https://medium.com/@dkn22/simplifying-embeddings-with-embedder-72bbb1eb22b3
=> embeder angucken

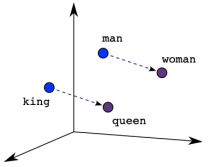
Entity Embeddings

Entity Embeddings
- of a categorial variable
- maps categorical variables into Euclidean space by solving a function approximation problem
- mapping learned by neural network during supervised training process
- Entity Embedding
  - are fixed-size continuous vectors that represent a categorical variable numerically
  - one unique vector for each category of variable
  - Embedding matrix: stack of these vectors
- Embedding layer:
  - maps each element in a set of discrete things - eg. words, users, or movies
  - to dense vector of real numbers - eg. its embedding

Small multi-dimensional space
- is large enough to because underlying data is usually very sparse
- Goal:
  - Enough dimensions: to encode relevant semantic relations
  - Small enough: to allow for good generalizations & efficient training

=> meaningful space enabling ML system opportunity to detect patterns

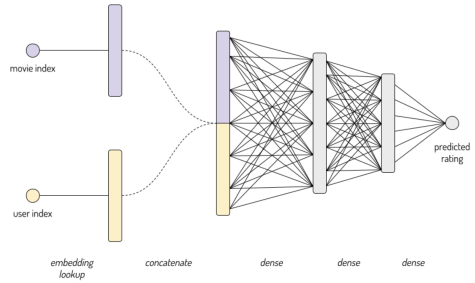
Position
- **Position** = distance and direction
  in vector space **encode semantics** in the embedding



Word Embeddings
- used to map words and phrases into a continuous distributed vector in a semantic space
- in semantic space:
  - similar words are closer
  - direction of the difference of vectors are meaningful - eg.
    King – Man ≈ Queen – Woman
    Paris – France ≈ Rome – Italy
- Learning word embedding by:
  1. Using word context with aim to maximize $p(w_c \mid w)$, where
     - w, w_c are vector repr. of a word w and its neighbor word w_c inside the context window
     - $p(w_c \mid w)$ probability to have w_c in the context of w
  2. Supervised methods - eg. learned by using text with labeled sentiment
     => similar approach used for categorial variables

Embedding as lookup-table

Embedding as lookup-table
- EE can be seen a lookup table
- embedding layer takes index as input
  eg. Titanic = 2543 => [.1,.4,-.3,.2]



Lookup of bag of words vector
- embedding is matrix
  - column = vector corresponding to an item (in vocabulary)
    => get dense vector for single vocabulary item - retrieve corresponding column
- Problem: How to translate sparse bag of words vector
- Solution:
  - get dense vector for a sparse vector representing multiple vocabulary items
    eg. all the words in a sentence or paragraph
  - retrieve embedding for each individual item
  - add them together
  - if sparse vector contains counts of the vocabulary items
    - before adding: multiply each embedding by the count of its corresponding item

  => same as matrix multiplication => embedding lookup as matrix multiplication
- Matrix multiplication: Given
  - 1 X N sparse representation S
  - N X M embedding table E
  - Matrix multiplication: S X E results in 1 X M dense vector

Functional API
- embeddings need functional Keras API
- to treat user and the movie as separate inputs,
  which come together only after each has gone through its own embedding layer.

Creating Embeddings

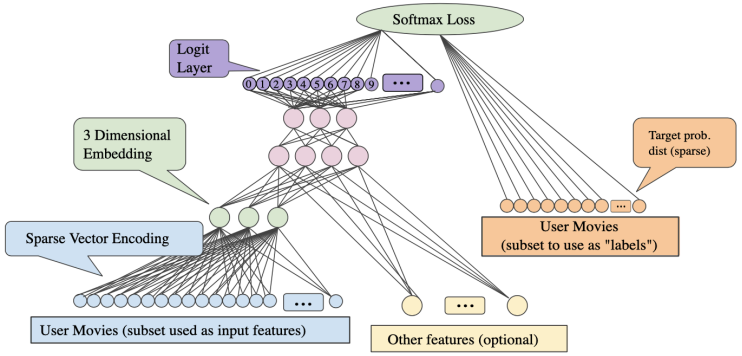
1. **Standard Dimensionality Reduction Techniques**
- PCA
  - given a set of instances like bag of words vectors
  - PCA tries to find highly correlated dimensions
    that can be collapsed into single dimension
- Singular Value Decomposition
  - normal SVD from linear algebra

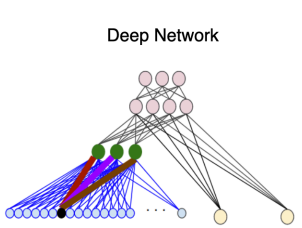
2. **Matrix Factorization**
- UV-Decomposition
- as used in Netflix-price

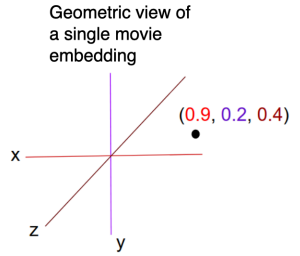
3. **Word2Vec**

4 **Creating Entity Embedding approach**
- using supervised learning task => „Creating entity Embedding using NN“

5. **Training an Embedding as Part of a Larger Model**
- similar to 4.
- if there is sparse data or dense data that should be embedded
- create an embedding unit => just special type of hidden unit of size d
- embedding layer can be combined with any other features and hidden layers
- as usual: final layer will be the loss that is being optimized
- Example: predict a user's interests from the interests of other users
  - modeled as a supervised learning problem
  - holding out small number of movies that the user has watched as the positive labels
  - optimize a softmax loss







Creating entity Embedding using NN

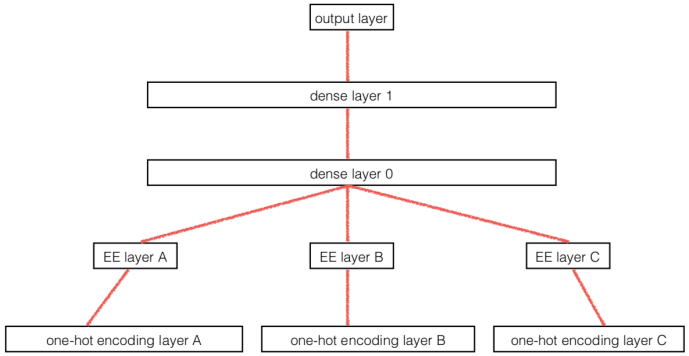
Entity Embedding
- Goal: put similar values of a categorical variable close to each other in embedding space
- How:
  - map discrete values to a multi-dimensional space
  - where values with similar function output
  - are close to each other in that space

EE in NN
- to learn the approximation ML function create mapping

- Mapping:
  - map each state of a discrete variable to a vector $e_i : x_i \rightarrow X_I$
  - mapping is equivalent to extra layer of linear neurons on top of the one-hot encoded input
    => **entity embedding layers equivalent** to **extra layer** on top of **each one-hot encoded** input

- Outputs of the embeddings layers:
  - are then concatenated
  - merged into a dense layer that served as input for the actual neural network

Embedding layer forces neural network to learn intrinsic properties of cat-variables



One-hot encoding of x_i:

$$u_i : x_i \rightarrow \delta_{x_i\alpha}$$

- $\delta_{x_i\alpha}$ is kronecker delta and possible values for α are the same as x_i
- if m_i = number of values x_i, then $\delta_{x_i\alpha}$ = vector of length m_i
  where the element is only non-zero when $\alpha = x_i$

Output of Embedding Layer

= output of extra layer of linear neurons given the input x_i

$$X_i = \sum_{\alpha} w_{\alpha\beta}\delta_{x_i\alpha} = w_{x_i\beta}$$

$w_{\alpha\beta}$ = weight connecting the one-hot encoding layer to embedding layer

β = index of the embedding layer

==> **mapped embeddings** are just the **weights of this layer** and are learned in same way as parameters of other layers

EE learns intrinsic properties of categorial variables
- network trained using back-propagation
- thus:
  - EE-layer learns about intrinsic properties of each category
  - deeper layers on top form complex combinations of them

Hyperparameter: Dimension of the embedding layer
- Hyper-parameters D_i = dimensions of the embedding layers
- choosing D_i
  - between 1 and $m_i - 1$. where m_i = number of values for the categorical variable x_i
  - based on experimentation
  - Guideline
    A. Start with intuition: how many aspects are needed to describe the entities
    B. No Intuition: start with $m_i - 1$

Using learned entity embeddings as input for other ML-algorithms
- EE learns intrinsic properties of categorial variables
- Idea: use entity embedding instead of one-hot encoded as input for other ML algs
  => improves their performance
- Why:
  - reduces noise: a form of dimensionality reduction
  - intrinsic properties: are revealed in this form of representation
  - linearity: introduces linearity by using other represenation
  - generalization: understanding intrinsic properties allows to better infer value when faced with new combinations of the variables

method	MAPE	MAPE (with EE)
KNN	0.290	0.116
random forest	0.158	0.108
gradient boosted trees	0.152	0.115
neural network	0.101	0.093