EMBEDDINGS

E.g. collaborative filtering:

Input: 1mil movies watched by 500k users
Task: Recommend movies to users

→ Need to determine movie similarity

1D of similarity is insufficient to properly capture this
2D also probably not (nice for visualization though)

→ Similarity = dist. between nodes in the D-dim. space you embed them into

But now you need to compute the embeddings, i.e. compute the D-dim. vectors.

How? Neural net, supervised by "user watched"

Representation?

- Per-user vec $\langle 1, 0, \ldots, 1 \rangle$ encoding whether user watched? Slow + big input
- Sparse encoding: list of movie IDs (but see below → this is just an optimization for your data)

Applications: ① Predict home sales price and use the home description text as an input to the regression model (eg. a NN)

So we need to learn an embedding layer to produce D-dim. vectors, then use that as inputs to the regression model.

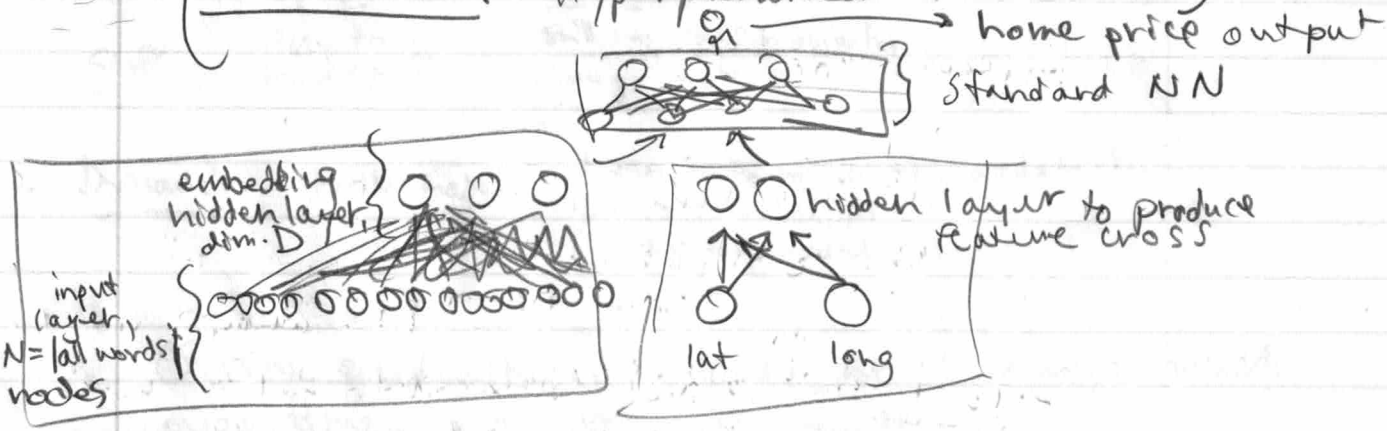
① Handwriting recognition
→ Learn embedding for the raw bitmap

③ Movie recommendations
→ Learn embedding for user movies sparse encoding

How to learn it?

① Just add a hidden layer (of dim. D nodes) in the NN between the input (of dim | vocab size | nodes) and the rest of the model! Back propagation will just "learn" it!? i.e. in passing.

(What is D? Hyperparameter. Tune it.)



→ Backpropagation just figures out everything.
- Feature crosses
- Embeddings
- Weights to combine them, etc.

(2) Standard dimensionality reduction methods (e.g. principal component analysis)

(3) word2vec: similarity = semantic based on word colocations. Train model.
(Invented at Google)
(See tutorial at tensorflow.org/tutorials/text/word2vec)