More on Word Vectors

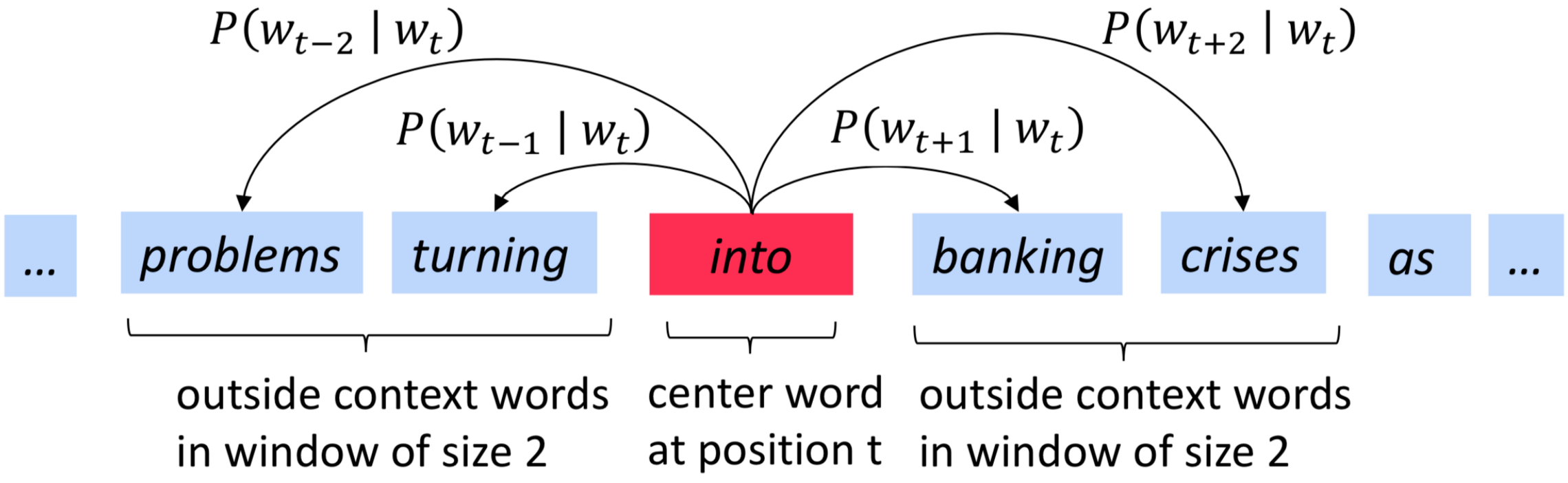
Lecture 3 Part 2

Today

- Wor2Vec recap
- Training the model
- Exploring the quality of our model
- Visualising the embedding in 2D
- Training a Sentiment Classifier
- Maybe a lecture about Habidatum

Recap

Skip-gram model



$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Credit: Stanford CS224n

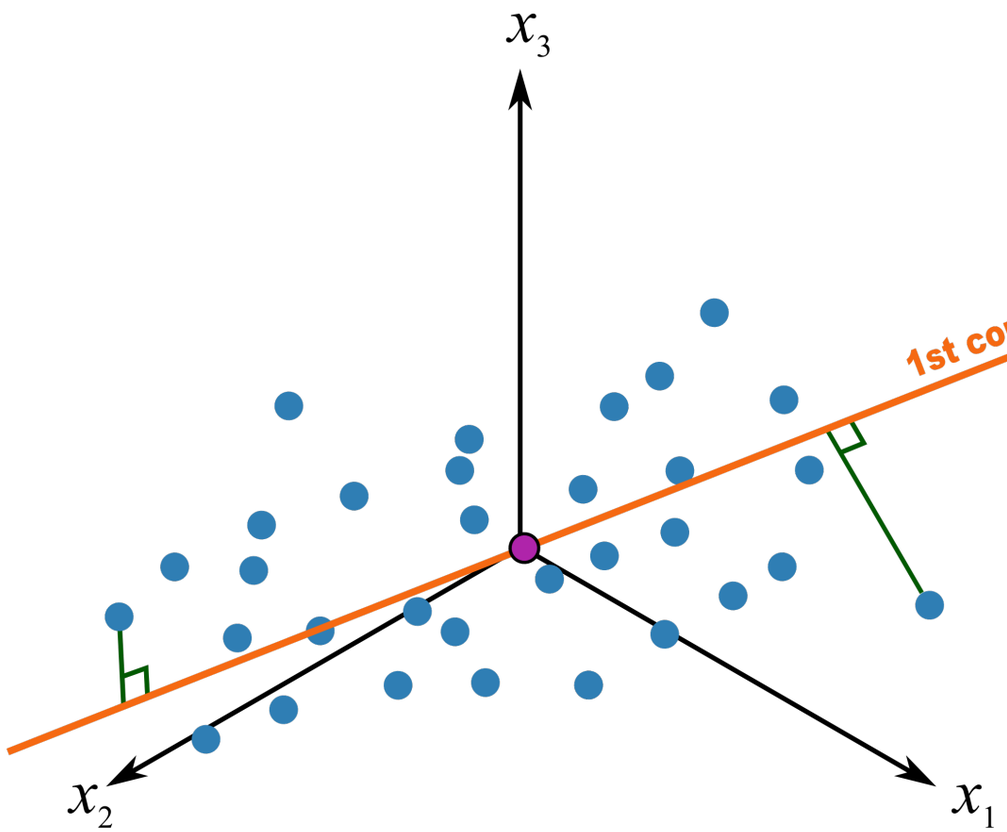
Evaluating Word Vectors

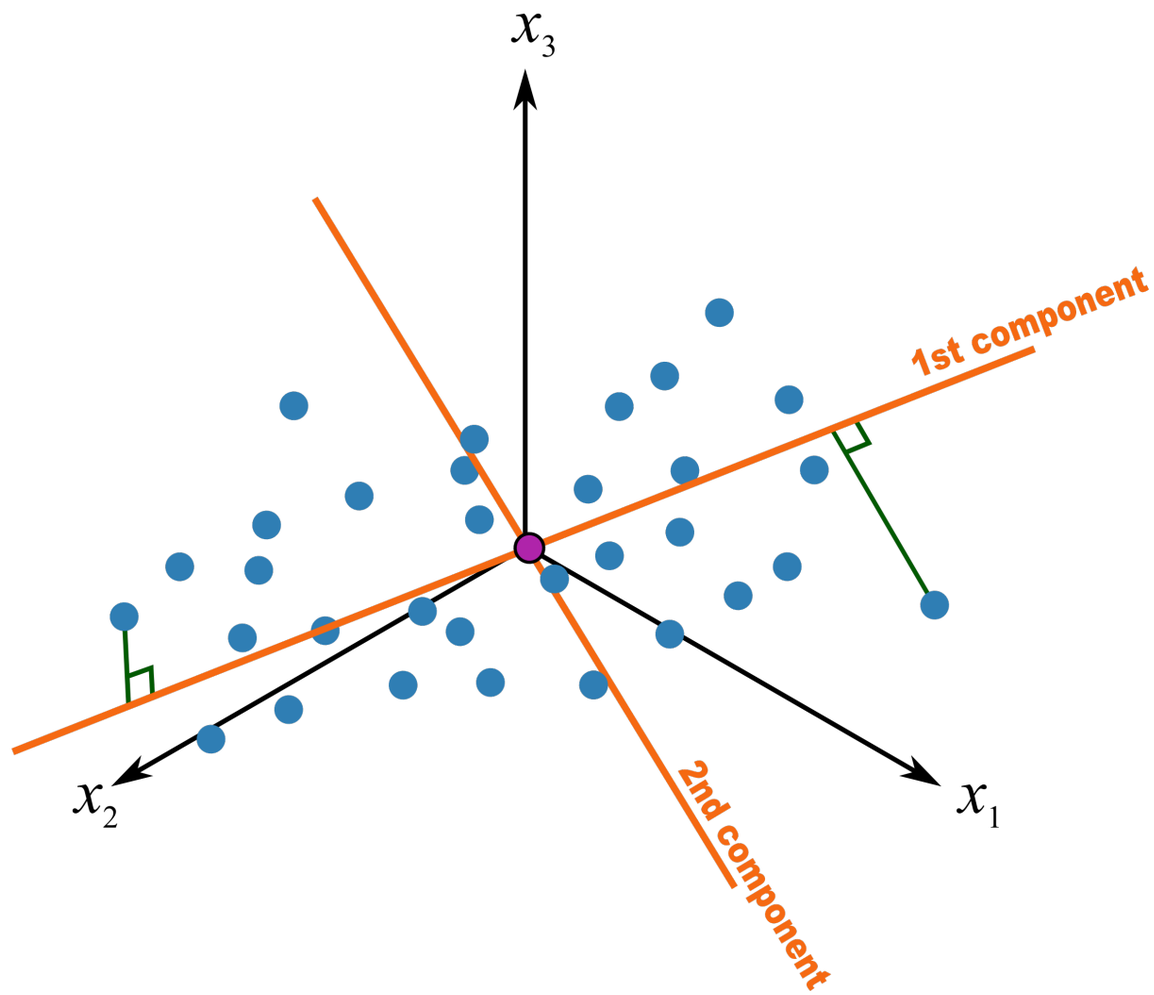
- Intrinsic evaluation: do word embeddings make sense?
    - Relatedness
    - Analogy
    - Categorisation
- Extrinsic evaluation: do word embeddings help solve a specific task?
    - Sentiment classification
    - Document classification

Dimensionality reduction

- Word vectors are still quite big: 128 coordinates
- How do we see what's going on inside?
- We need to make a projection to a space with fewer number of dimensions (2 is best)

Dimensionality reduction: PCA





Dimensionality reduction: PCA

- There are many dimensionality reduction techniques, most common: PCA, Primary Component Analysis

- It tries to find some new coordinate space, with axis as combinations of original ones, trying to explain the most variance of the data

- Computable analytically, but takes a lot of time on large data and is limited in the way it can reduce the dimensions

- More on it in other lectures

Dimensionality reduction: t-SNE

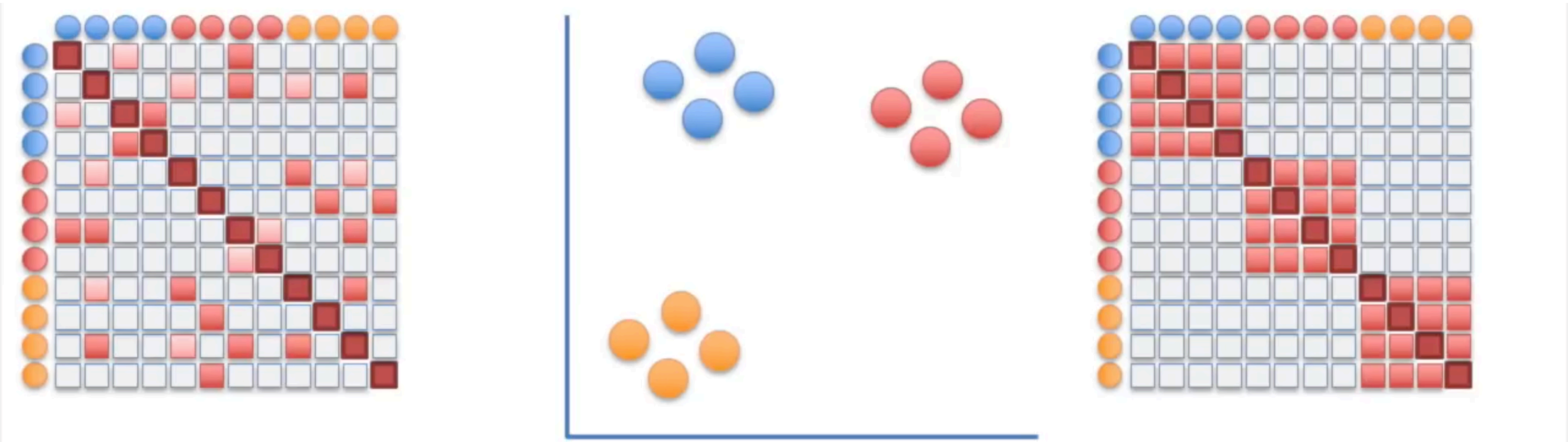
- t-SNE: one of unsupervised way to learn a dimensionality reduction model, non-linear, unlike PCA. *Developed by Laurens van der Maaten*

- The general idea is to make a map between old points and new ones such that the distance between close points will be close, and distance between distant would be distant

- t-SNE stands for t Distributed Stochastic Neighbor Embedding. So there is t-Distribution of distances in the new space

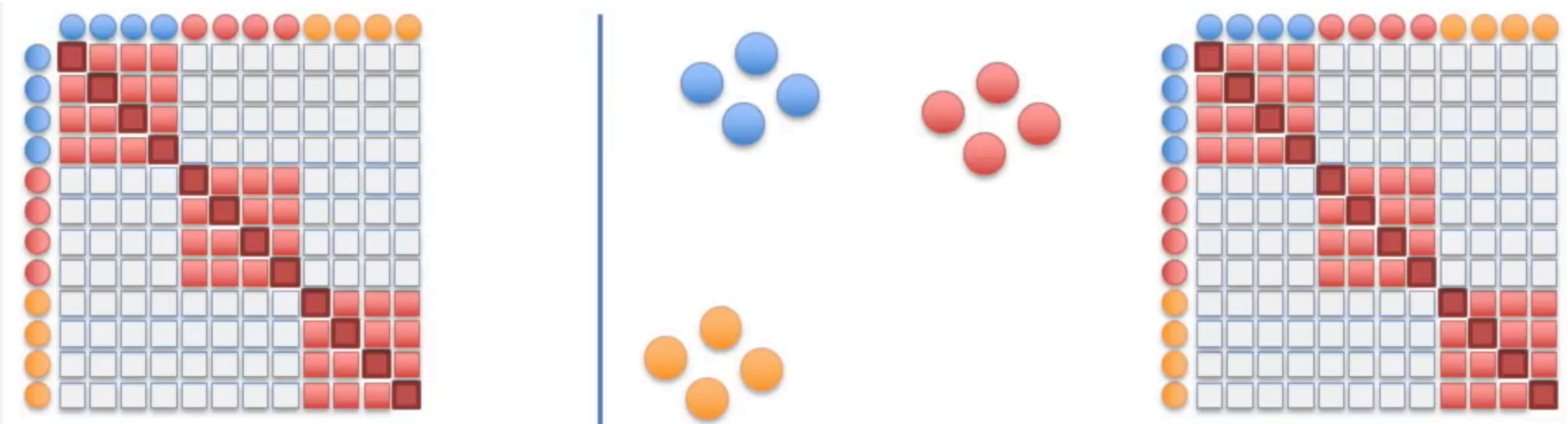
- Can’t easily include new data into it, needs to relearn

Dimensionality reduction: t-SNE



t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

Dimensionality reduction: t-SNE



t-SNE moves the points a little bit at a time, and each step it chooses a direction that makes the matrix on the left more like the matrix on the right.

Two similarities matrixes

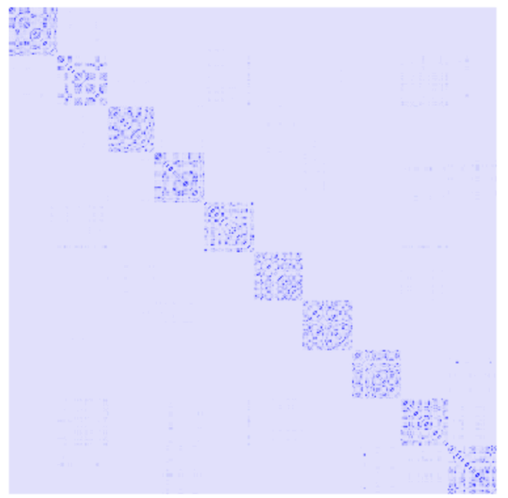
Original $p_{j|i} = \frac{\exp\left(-|x_i - x_j|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-|x_i - x_k|^2 / 2\sigma_i^2\right)}$ **On a 2D map** $q_{ij} = \frac{f(|x_i - x_j|)}{\sum_{k \neq i} f(|x_i - x_k|)}$ with $f(z) = \frac{1}{1+z^2}$

- *P* is fixed, but we need to adjust *Q*.

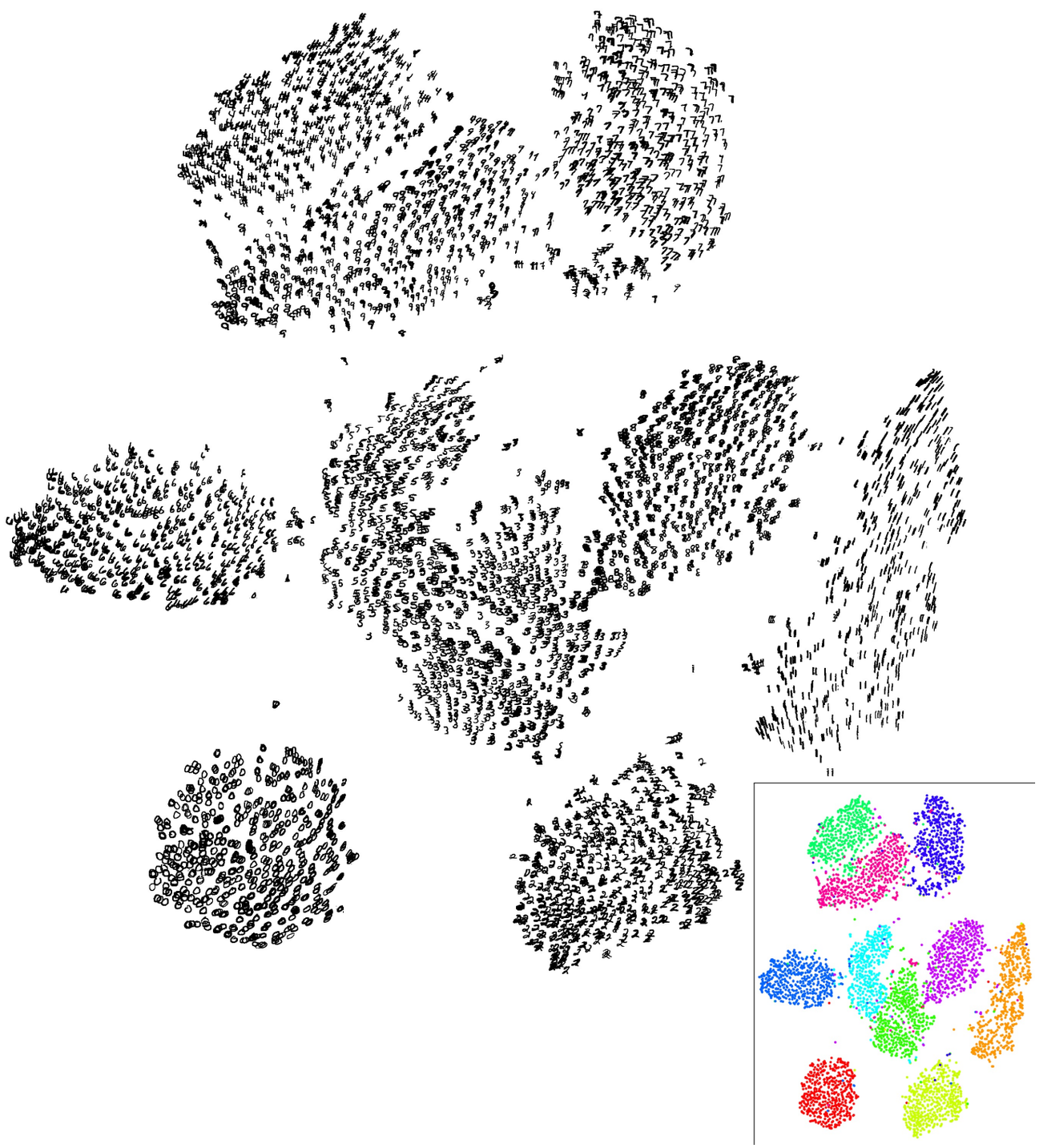
- We want to make them similar. How?

- Optimising the KL divergence between two distributions of distances

$$KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$



How it looks with MNIST



Using word vectors

- Let’s see what we can do to make sentences from words

- Let’s learn a classification task

Sentiment classification

- Predict the sentiment of a given tweet

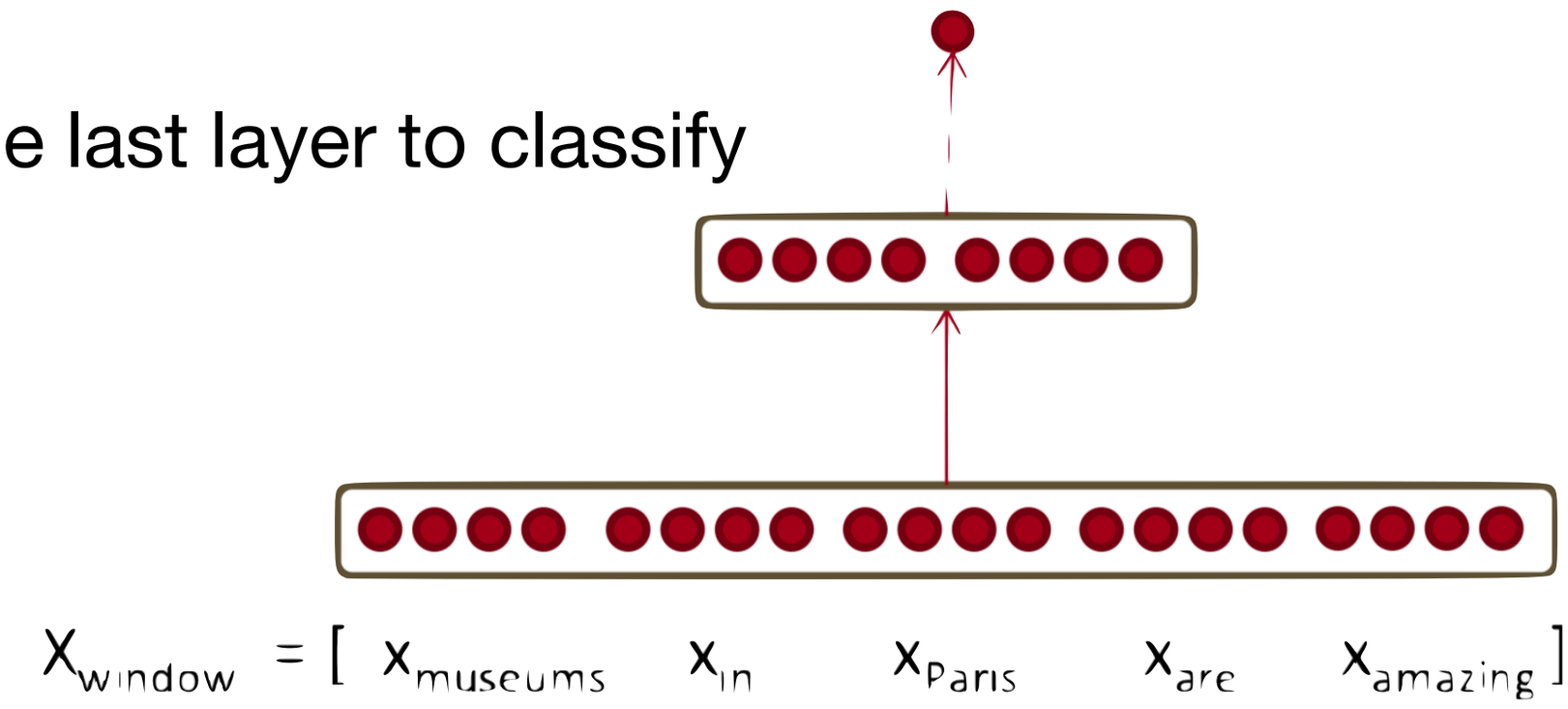
- Compare BOW and LinearSVC with Deep NN on word vectors

Linear SVC

- Treat each vector dimension as a feature/column in our dataset
- Make and BOW model: sum or average all words' vectors in the sentence
- Pass a training dataset to a classifier: let's use Linear Support Vector Classifier

Neural Net for Classification

- Let’s concat words vectors for a sentence vector
- Non-linear layer will let us learn some dependency between word sequences as features
- Use sigmoid at the last layer to classify



Tuning word vectors

- In classical ML we don’t change our inputs
- With word vectors, we actually can do that
- Have to be careful and not to overfit, so better use some regularisation for sure
- And only do that given a big enough dataset
- Otherwise, use the vector you have already