Representation Learning. Teaching machines to learn features

Big Picture:

- Representation learning. Learn useful features for downstream tasks.

What are features?

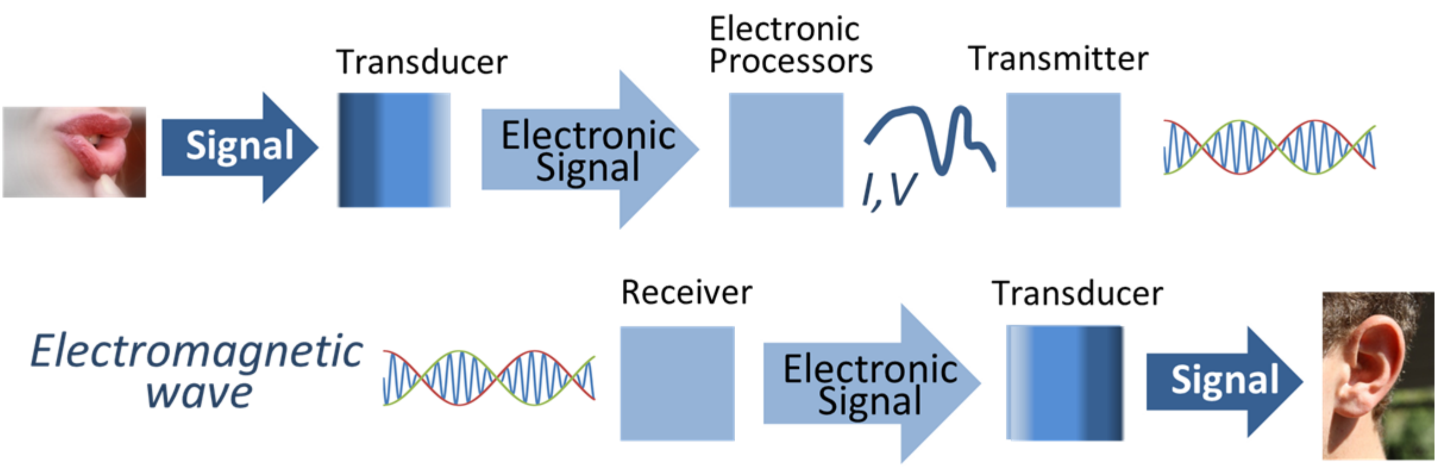
- first examples
- then provisional definitions

Expert designed features

- Hand made features depending on the type of data, defined by an expert.
- E.g. functions of $f(t)$ to learn. Interpolate with special functions (sin/cosine/fourier). Data, wavelets

Signal processing

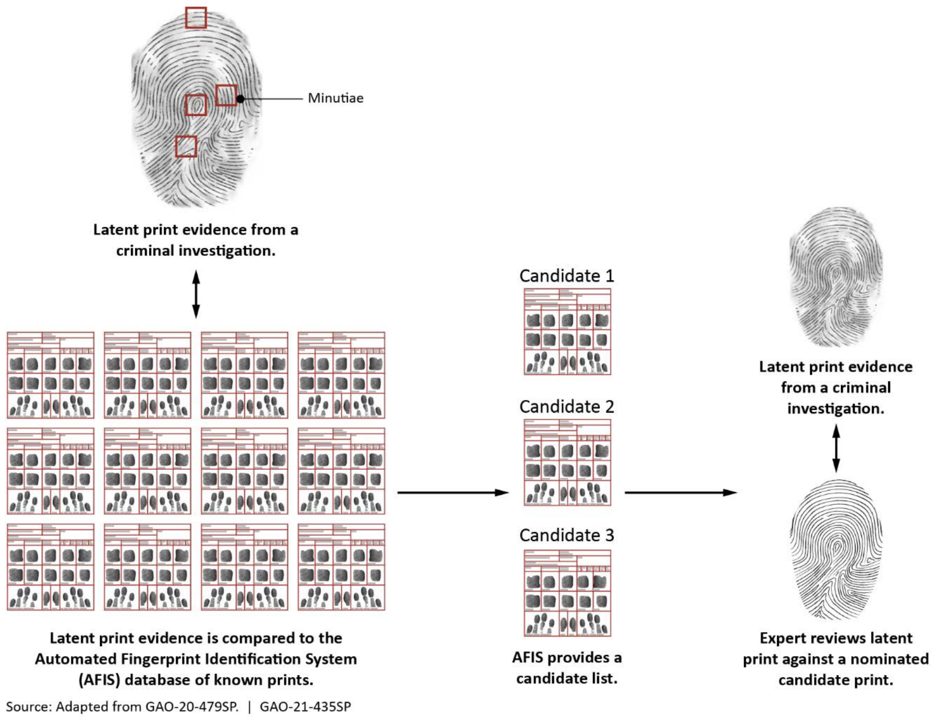
https://commons.wikimedia.org/w/index.php?curid=19048816



Handmade features: e.g. Fingerprints

- By U.S. Government Accountability Office from Washington, DC, United States - Figure 1: The workflow for latent print analysis for law enforcement investigations, Public Domain, https://commons.wikimedia.org/w/index.php?curid=110178216

Figure 1: **The workflow for latent print analysis for law enforcement investigations**



Provisional definition of features

- raw data x in n-dimesions high dimensional - otherwise can use the raw data as features.
- raw data: may not be possible to learn from it - may not be a vector, e.g. finger print.
- features $f(x)$ in d-dimensions (smaller than x)
- Feature is useful for a downstream task (classification) (expected loss minimization from S_m) if
  - 1. not possible to learn directly from raw data. But can learn from features.
    2. can learn more effectively from features than from raw data.
- (later extend to other tasks)
- If the shallow classifier we learn using $f(x)$ is a better fit to the data than the one we learn from x (i.e. lower loss), using the same loss, hypothesis class. (but different)

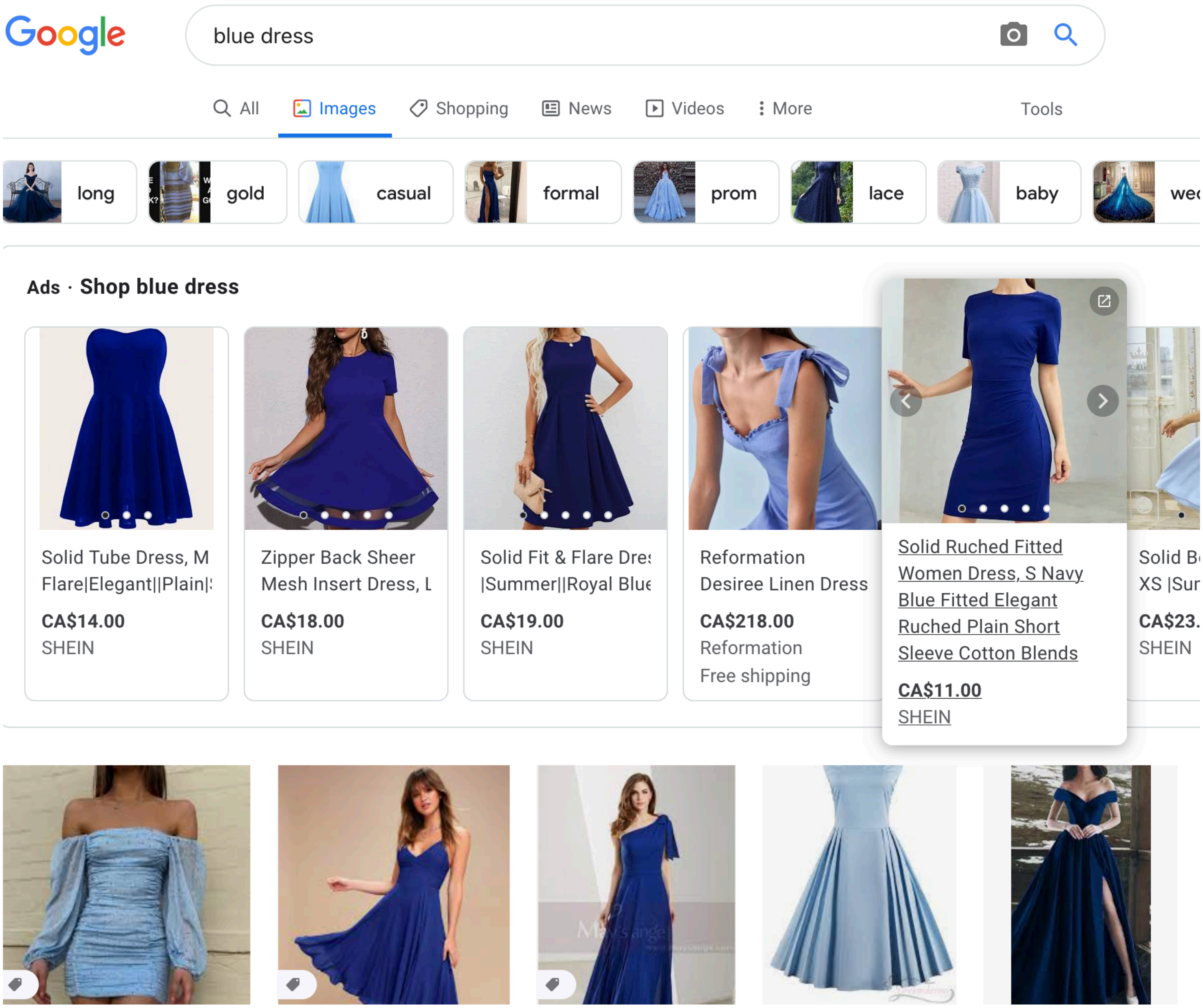
Trivial features: watermark/bar code

- each image has an embedded bar code, that can be converted into what we want.
- E.g. with phone.

Non-trivial features.

Machine learning features

- Feature representation. Can we *learn* features from data.
- Kernel features: "kernel trick" method of defining features from data. (Less important/later)
- Deep neural network features: Imagenet features



Using DNN features

- Similarity search.
-

Learning DNN features:

- Train a classifier, hypothesis deep neural network. (Deep neural network: will define later)
- Strip the last layer of the nextwork: this defined the features.

Similarity: In Image features

- Cosine feature similarity. Inputs: image x, feature $f(x)$ (deep neural network features). Output
- $s(x, x') = f(x) \cdot f(x')/|f(x)||f(x')|$ which is cosine of angle between vectors.
- Reverse Image search. Find similar images from a data base, given a sample image. Based
- Face Recognition. Classify two images as belonging to the same face (or object) based on feature similarity. Cosine similarity of $f(x)$. and $f(x')$.

Similarity for words: word2vec.

- Inputs: list of words (e.g. 20000 most common words). Corpus of text.
- Output: vector representation in d dimension of each word. Such that similar words are geometrically similar: dot products.
- Method: gather the "co-occurence": probability that two words appear nearby (log version).
- Define: similarity of two words :

Add to lecture notes:

- Losses: 0-1 Classification loss, not differentiable!

Losses: Image Segmentation.

- Now the input is x image with d pixels, output, d classes (one at each pixel).
- What is the loss? 0-1 loss on each pixel, average. Now average error. Not good if unbalanced (which is usually the case).
- More sophisticated error: Inside should be easy. Class boundaries are where it matters.
- Can you do a differentiable loss? Can be a multi-class margin loss.
- What about different weighting on the loss? Can be larger where: correct labels change. (i.e. gradients)
- Need to check if this makes sense?