Deep Learning (BEV033DLE)
Lecture 10
Learning Representations

Czech Technical University in Prague

Learning Representations Block

Discriminative

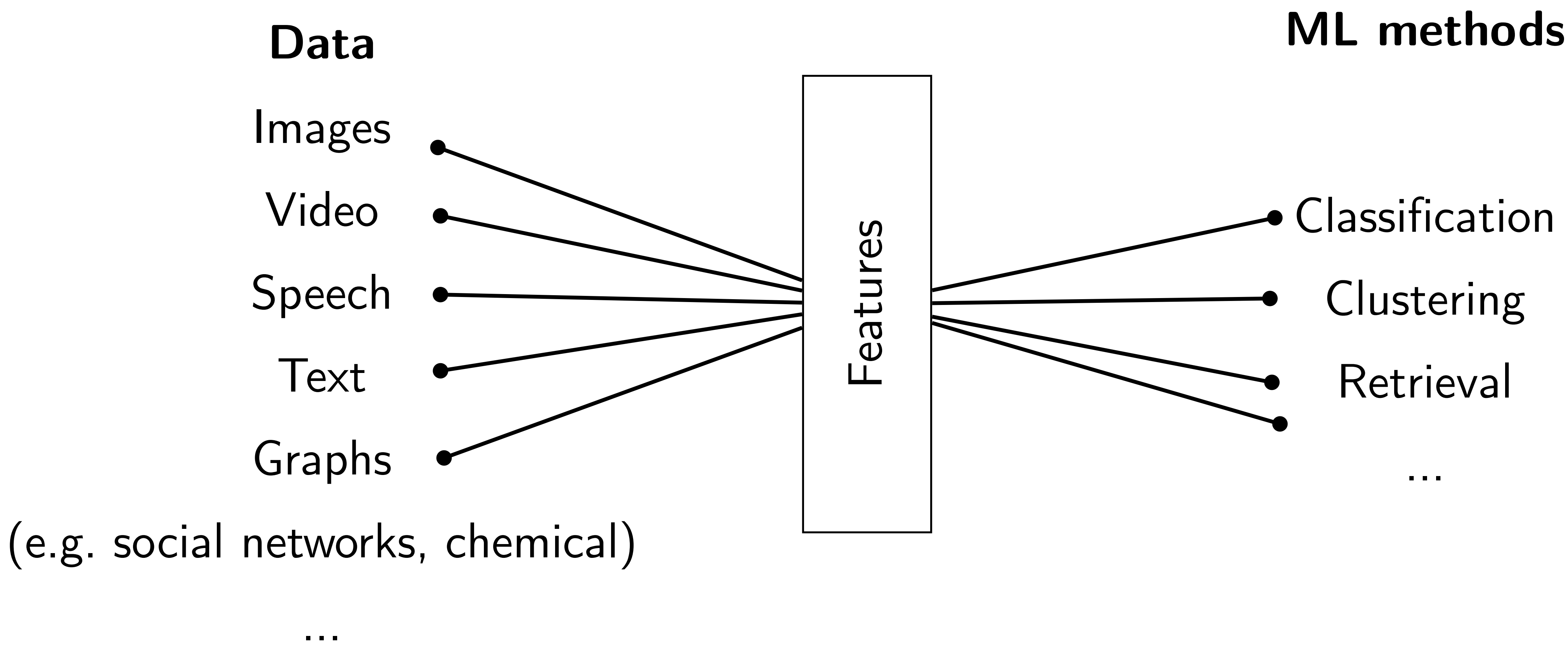
- Lecture 10: LR-1:
  - Representations
  - Similarity / metric learning
  - Word Vectors
- Lecture 11: LR-2:
  - Stochastic embedding: tSNE
  - KL Divergence
  - Latent Variable Models
    - Multi-sense word vectors
    - Stochastic EM, Variational inference
- Lecture 12: Variational Autoencoders

Generative

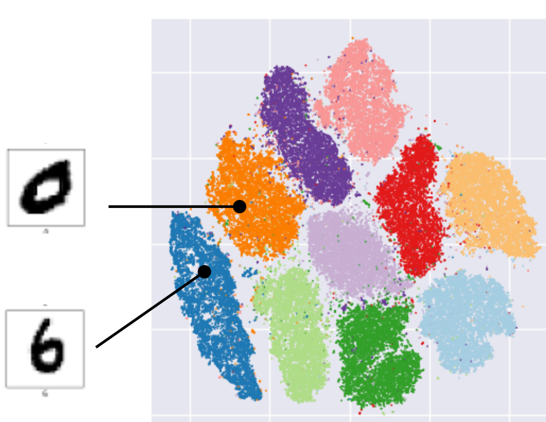
Representations

Features

✦ Conventional Machine Learning



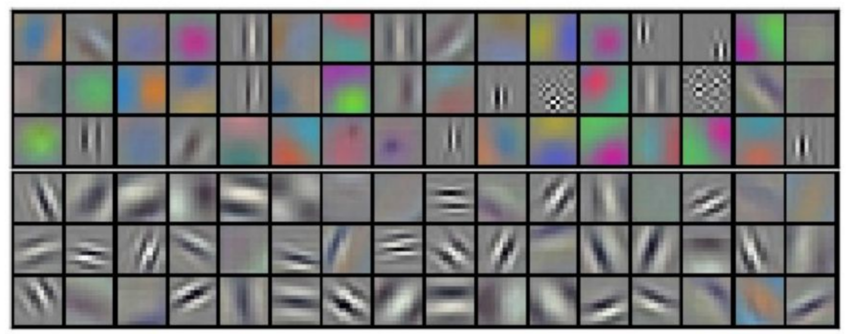
- With good features all ML tasks become simpler
- Features are engineered or selected



Features in Deep Learning

✦ Deep Learning

- Low level features in vision appear to be generic
- Even deep features appear to be useful via fine-tuning

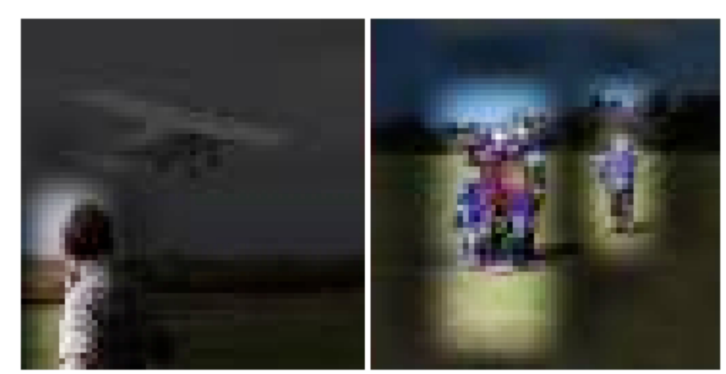


- In networks trained for different complex problems some intermediate layer activations correspond objects or object parts, while never explicitly supervised to do so

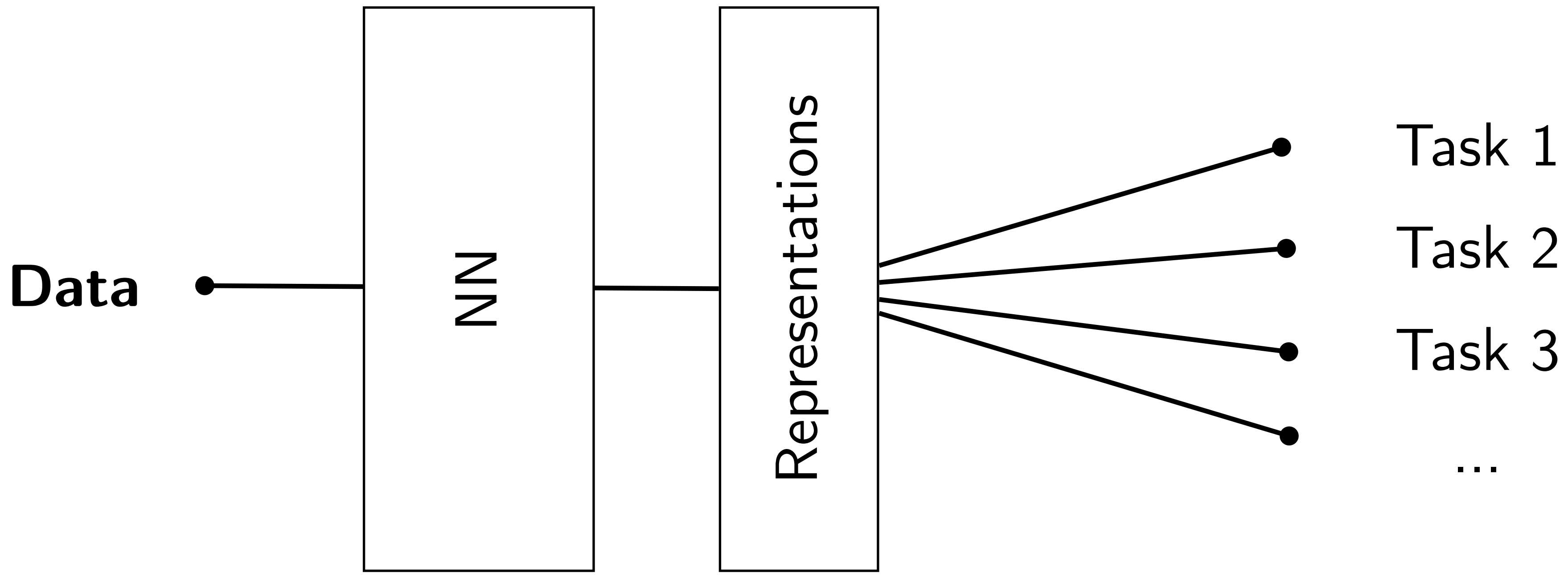
lamps in places net

wheels in object net

people in video net



Representation Learning



- Desire for good representations:
  - Keep useful information (useful for all real tasks)
  - Discard unnecessary information (invariant to view, lighting, etc.)
  - With topology and metric in the feature space
  - Learned from the data with or without human annotation
  - Interpretable
- Hard to formalize: e.g. discarding information restricts the scope of tasks

Similarity/Metric Learning

Similarity Learning

✦ Goals:

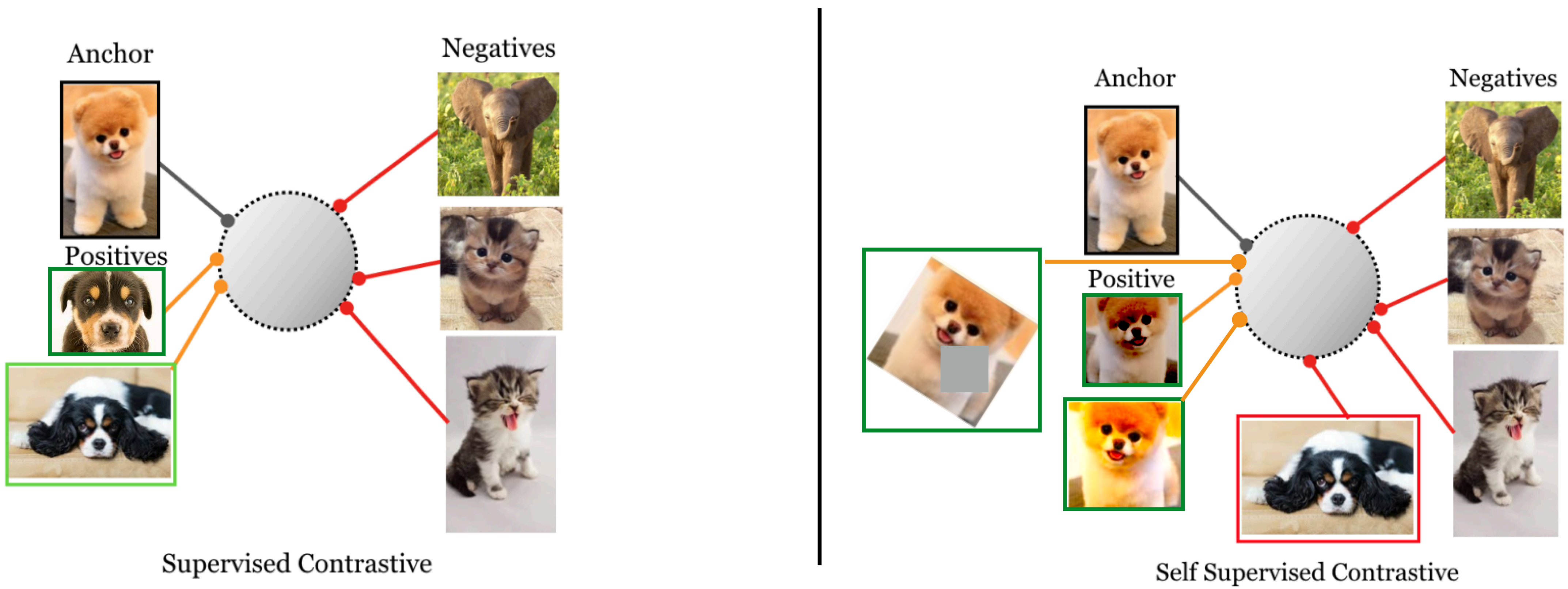
- learn the concept of **similarity** of two objects
- quantify this similarity

✦ Supervised learning:

- Given examples of "similar" and "distinct" pairs learn the function sim(x,y)
- no GT of the values of sim(x,y)

✦ Self-supervised learning:

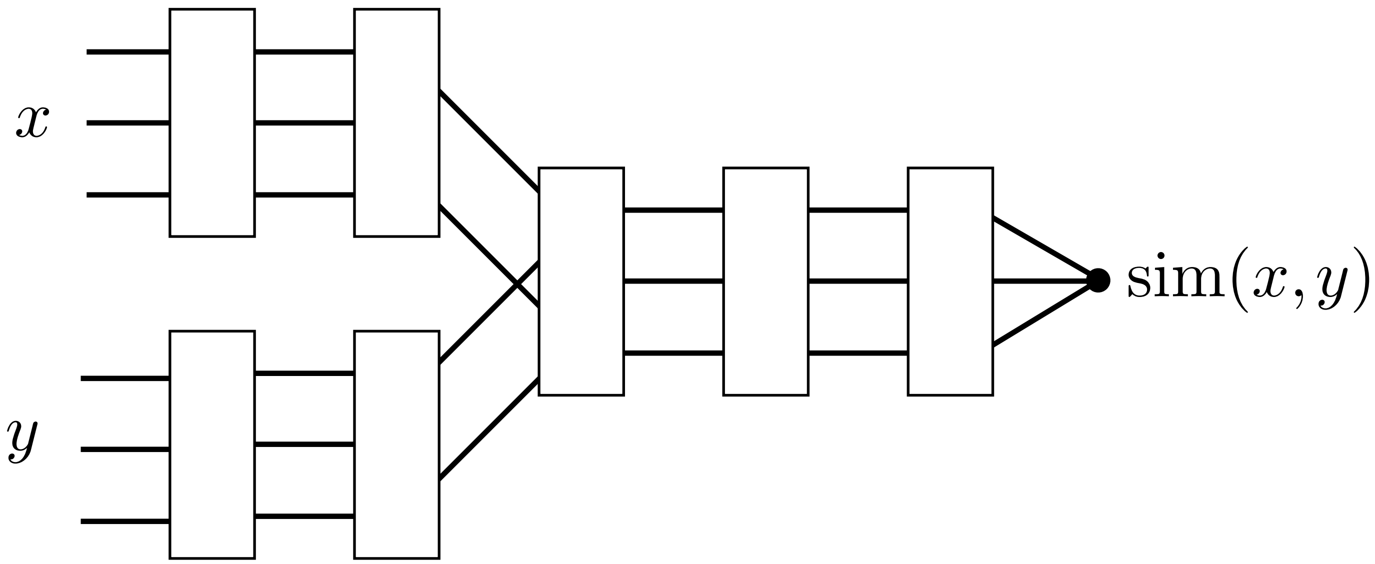
- Create "similar" pairs by identity-preserving transforms



[Khosla et al. (2020) Supervised Contrastive Learning] original graphics edited for visualization

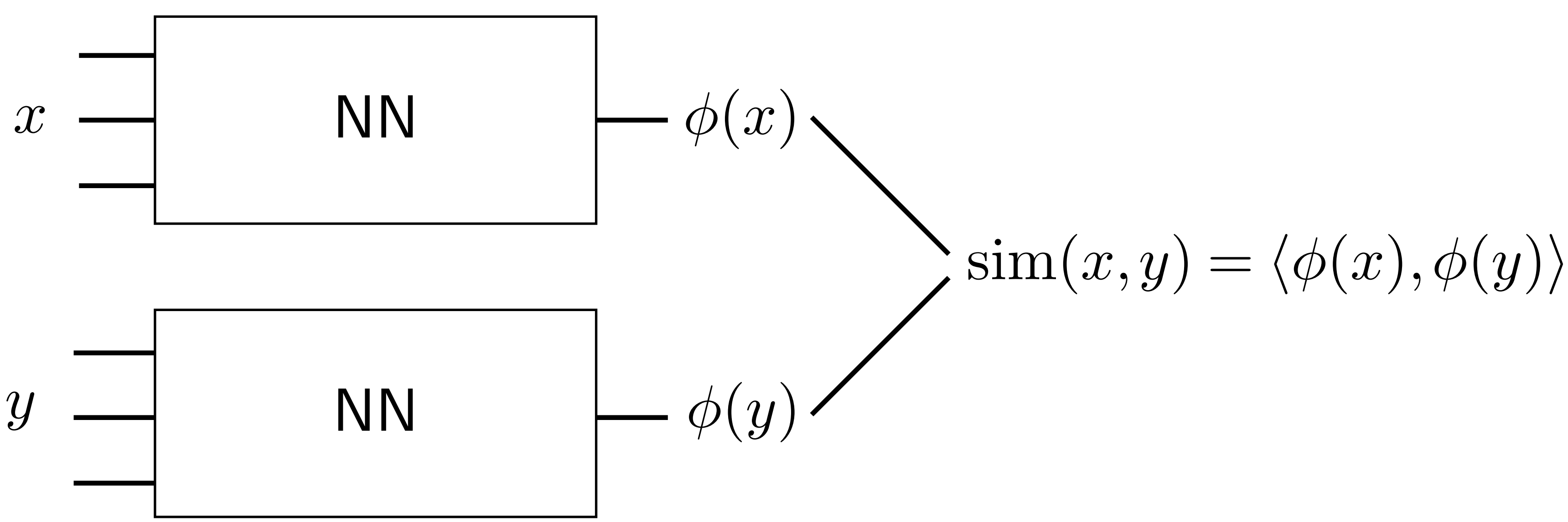
Model

- Approach 1: generic network with two inputs



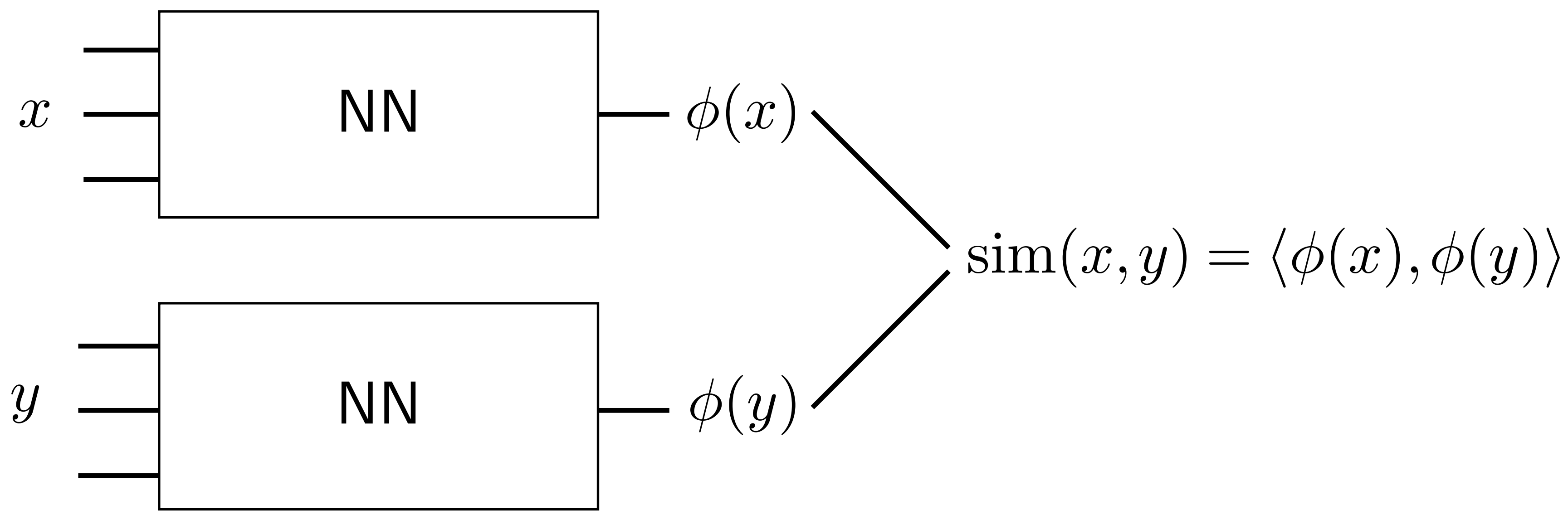
first layers extract generic features -- can be shared

- Approach 2: network creates representations (embeddings / features)



Model

✦ Network creates representations (embeddings / features)



- Inner product: $\mathrm{sim}(x,y) = \langle \phi(x), \phi(y) \rangle$
  Can represent any Kernel $K(x,y)$
  Maximum Inner Product Search (MIPS)

- Euclidean: $\mathrm{sim}(x,y) = -\|\phi(x) - \phi(y)\|^2$
  nearest neighbor search (NNS): sub-linear approximate methods

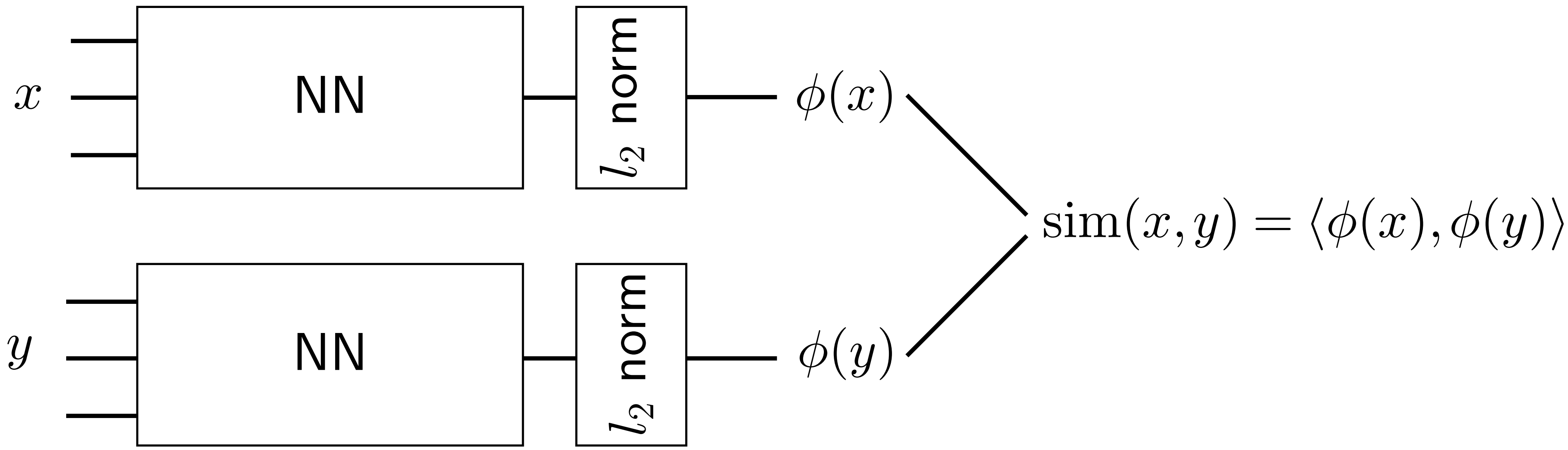
- Correlation: $\mathrm{sim}(x,y) = \frac{\langle \phi(x), \phi(y) \rangle}{\|\phi(x)\| \|\phi(y)\|}$ correlation-NNS: sub-linear approximate methods

◆ All equivalent if $\|\phi(x)\| = 1$ for all x

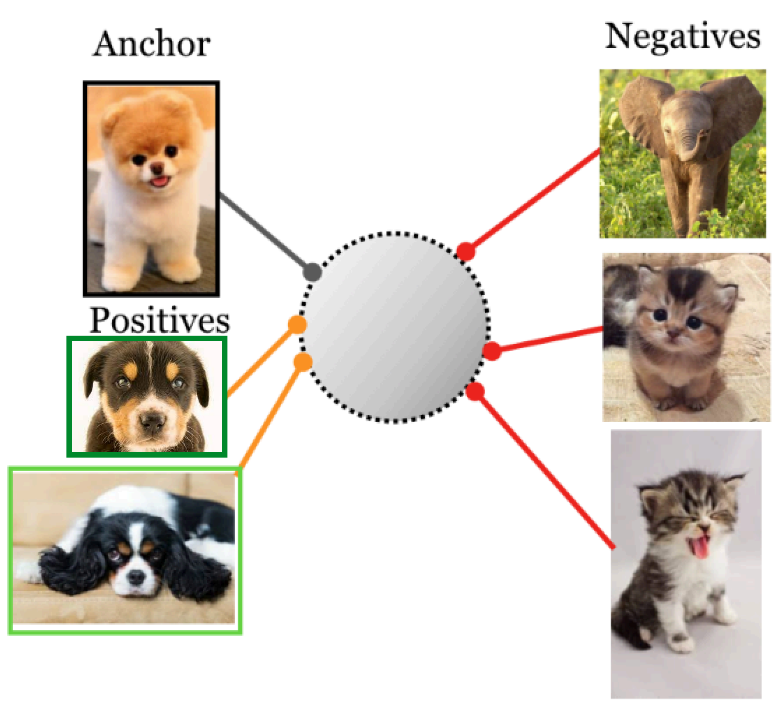
◆ There are known mappings to approximate $\langle u, v \rangle$ with $|\ P(u) - Q(v)\|^2$ or $\frac{\langle P(u), Q(v) \rangle}{\|P(u)\| \|Q(v)\|}$

Model

- Convenient common model: normalize representations (equivalent to using correlation for similarity)



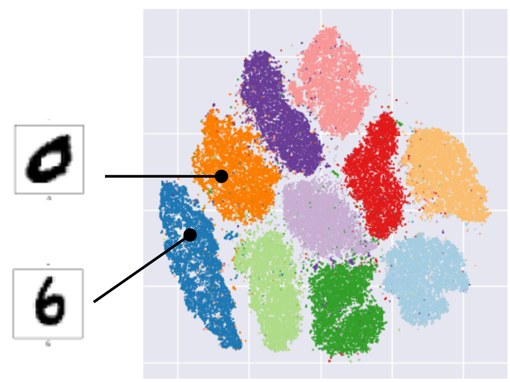
Representations live on a hypersphere



Applications

✦ Classification:

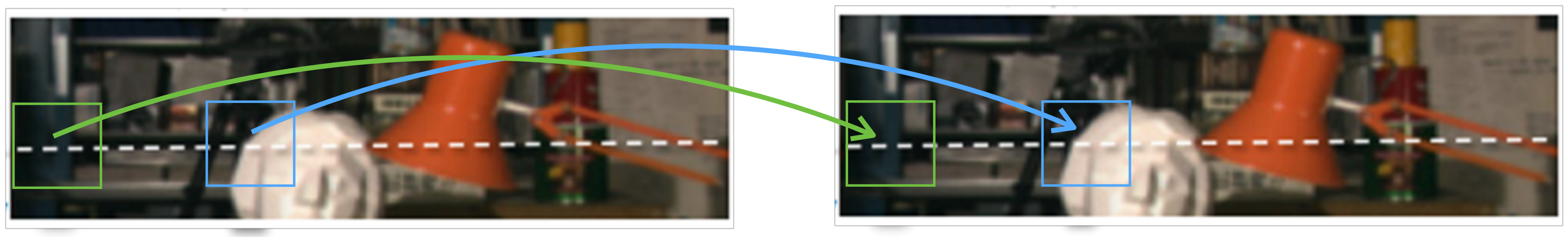
- Classify by NNs
- Learn classification head
- Fine-tune the whole model (same or different data)



✦ Visual search (retrieval):

- have a large database
- want to query with image or text
- retrieve similar objects or views

✦ Correspondence search (structure from motion, stereo, optical flow)

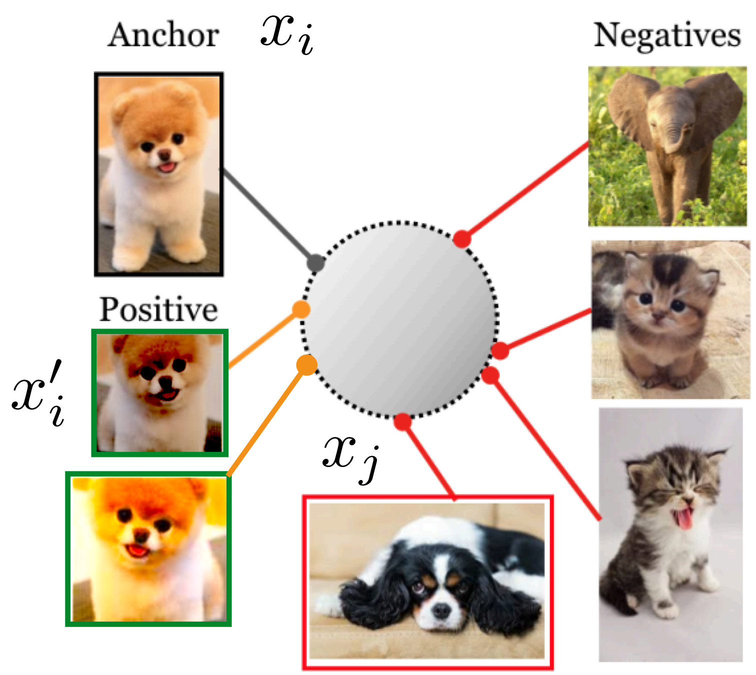


✦ Data exploration

[Johnson et al. (2017)]

Instance Classification

- Training data: $x_1 \dots x_N$
  - *Anchor*: x_i
  - *Positive*: $x' = T(x_i)$ – random transform
- Classifier:
  - As many classes as there are instances (data points)
  - Score of instance i: $s_i = \phi(x_i)^\mathsf{T}\phi(x')$
  - $p(y{=}i|x') = \frac{e^{s_i}}{\sum_j e^{s_j}}$
  - Same learning formulation as we used for classification
  - Large sum in the denominator $\rightarrow$ noise contrastive estimation
  - Ensures instances can be discriminated
  - Enforces invariance to transformations



Self Supervised Contrastive

Dosovitskiy et al. (2014): Discriminative unsupervised feature learning with convolutional neural networks

Wu et al. (2018): Unsupervised Feature Learning via Non-Parametric Instance Discrimination]

Chen et al. (2020): A Simple Framework for **Contrastive Learning** of Visual Representations

Contrastive learning: "contrasting positive pairs against negative pairs"

Triplet Loss

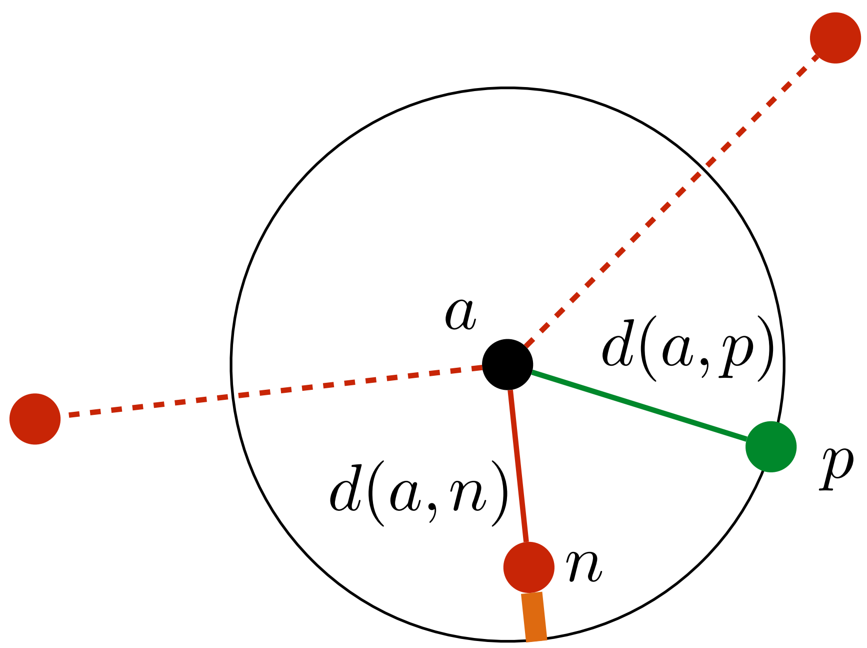
- Training data: $x_1 \dots x_N$
  - Positive pairs: $x_i \sim x_j$ for $(i,j) \in \mathcal{P}$
  - Negative pairs: $x_i \not\sim x_j$ for $(i,j) \in \mathcal{N}$
  - Example: known class label $\rightarrow$ similar if same class
- Desired separation property:
  - a – *anchor* (any data point)
  - p – *positive* sample for a
  - n – *negative*
  - let $D_p = \mathrm{d(a,p)}$ and $D_n = \mathrm{d(a,n)}$
  - Want:

$$D_p < D_n \qquad \forall p, n$$

$$D_p - D_n < 0$$

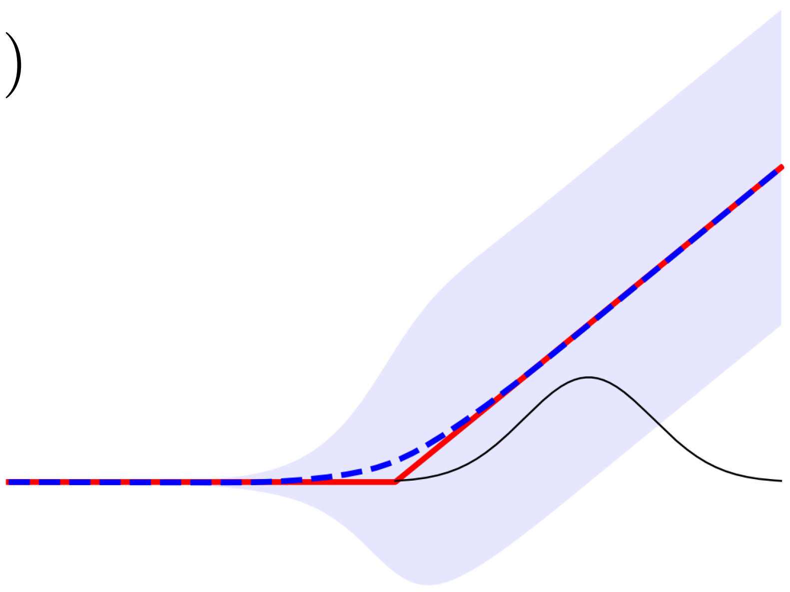
  - Hinge loss 1: $l = \sum_{n \in \mathcal{N}(a)} \max(0, D_p - D_n)$
  - Hinge loss 2: $l' = \max(0, \max_{n \in \mathcal{N}(a)} (D_p - D_n)) = \max_{x \in \mathcal{N}(a) \cup \{p\}} (D_p - D_x)$

  -- **hard negative mining**



Smooth Triplet Loss

- Hinge loss variant 1: $l = \sum_{n \in \mathcal{N}(a)} \max(0, D_p - D_n)$



Smooth maximum (SoftPlus):

$Z \sim \text{Logistic}(0, \tau)$

$\mathbb{E}[\max(0, x + Z)] = \tau \log(1 + e^{x/\tau})$

Let $\tilde{D}_p, \tilde{D}_n$ be noisy descriptors with additive noise $\text{Gumbel}(0, \tau)$

$\mathbb{E}[\max(0, \tilde{D}_p - \tilde{D}_n)] = \mathbb{E}[\max(0, D_p - D_n + Z)] = \tau \log(1 + e^{(D_p - D_n)/\tau})$

Used e.g. in [Sohn 2016]

- Hinge loss variant 2: $l' = \max_{x \in \mathcal{N}(a) \cup \{p\}} (D_p - D_x)$
  - Let $\tilde{D}_p$, $\tilde{D}_x$ be noisy descriptors with additive noises $\text{Gumbel}(0, \tau)$
  - $\mathbb{E}[l] = -\tau \log \frac{e^{-D_p/\tau}}{\sum_x e^{-D_x/\tau}}$
  - Log softmax loss back again
  - For the inner product similarity: $\mathbb{E}[l] = -\tau \log \frac{e^{\phi(a)^\top \phi(p)/\tau}}{\sum_x e^{\phi(a)^\top \phi(x)/\tau}}$

Often used in papers without explanation (sometimes called contrastive loss)

Word Vectors

Word Vectors

- Example: Simple model for predicting context words:
  - Assume a finite vocabulary I, $|I| = n$
  - For every word x in the text, try to predict all nearby words y

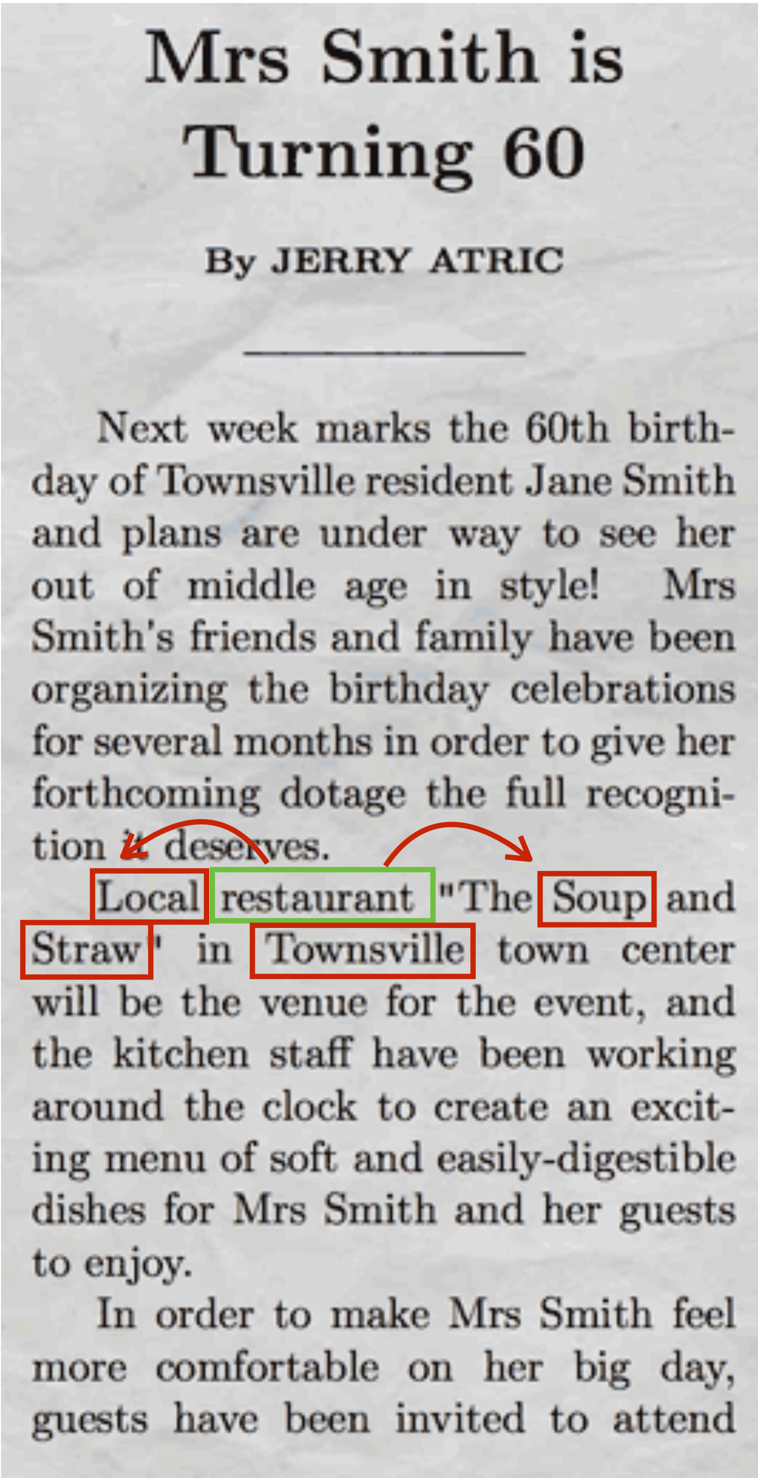
- Model (predictive coding):
  - Two embeddings:
    anchor $v(x) \in \mathbb{R}^d$ and *context*: $u(y) \in \mathbb{R}^d$
  - All embeddings are fully specified by matrices U, V of size $n \times d$ – no deep network.
  - Categorical conditional distribution:

$$p(y|x) = \frac{\exp(u(y)^{\mathsf{T}} v(x))}{\sum_{y'} \exp(u(y')^{\mathsf{T}} v(x))}$$

  - Learn by maximum likelihood:

$$\max_{U,V} \prod_t \underbrace{\prod_{t' \in \mathcal{N}(t)} p(y_{t'}|x_t),}_{\text{Naive Bayes model}}$$

t – position in the text, $\mathcal{N}(t)$ – nearby positions



Mrs Smith is Turning 60

By JERRY ATRIC

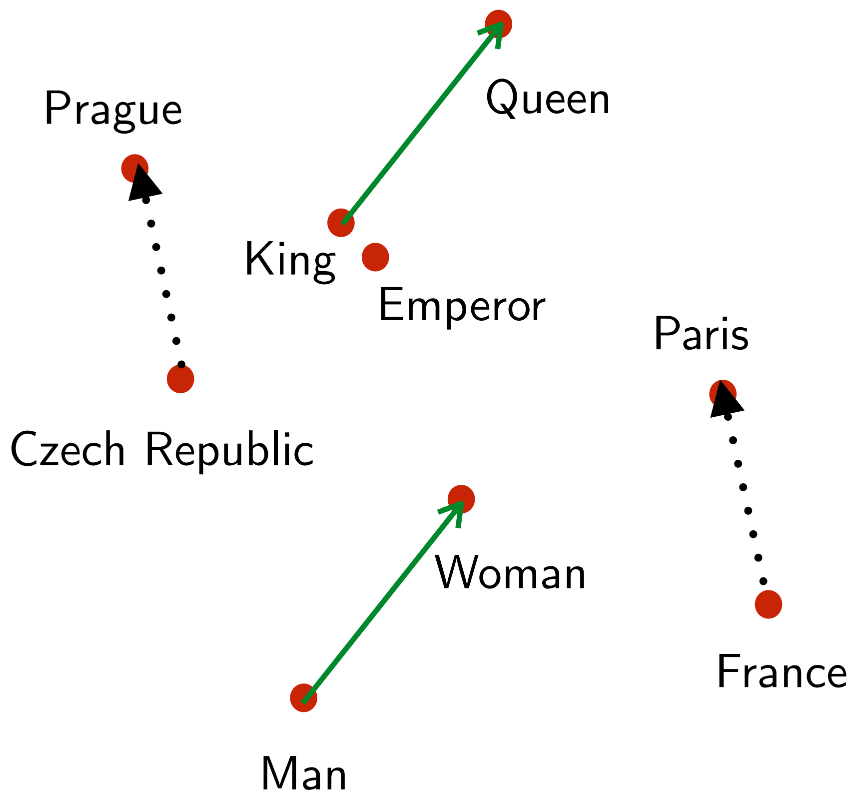
Next week marks the 60th birthday of Townsville resident Jane Smith and plans are under way to see her out of middle age in style! Mrs Smith's friends and family have been organizing the birthday celebrations for several months in order to give her forthcoming dotage the full recognition it deserves.

Local restaurant "The Soup and Straw" in Townsville town center will be the venue for the event, and the kitchen staff have been working around the clock to create an exciting menu of soft and easily-digestible dishes for Mrs Smith and her guests to enjoy.

In order to make Mrs Smith feel more comfortable on her big day, guests have been invited to attend

Word Vectors

- Learned a **representation** of each word x as the embedding $v(x) = V_{x,:} \in \mathbb{R}^d$



- Direction of $v(x)$ appears to capture abstract relations:
  - Semantic:

    `"King" - "Man" + "Woman" ≈ "Queen"`

    `"Prague" - "Czech Republic" + "France" ≈ "Paris"`

    `"Czech" + "currency" ≈ "koruna"`
  - Syntactic:

    `"quick" - "quickly" ≈ "slow" - "slowly"`
- Evaluated on a corpus of relation prediction tasks
- More complex tasks become easier when using such vector representations