Learning with Small Samples

Including zero-shot learning

Nour Karessli
DSR 2019

Structure

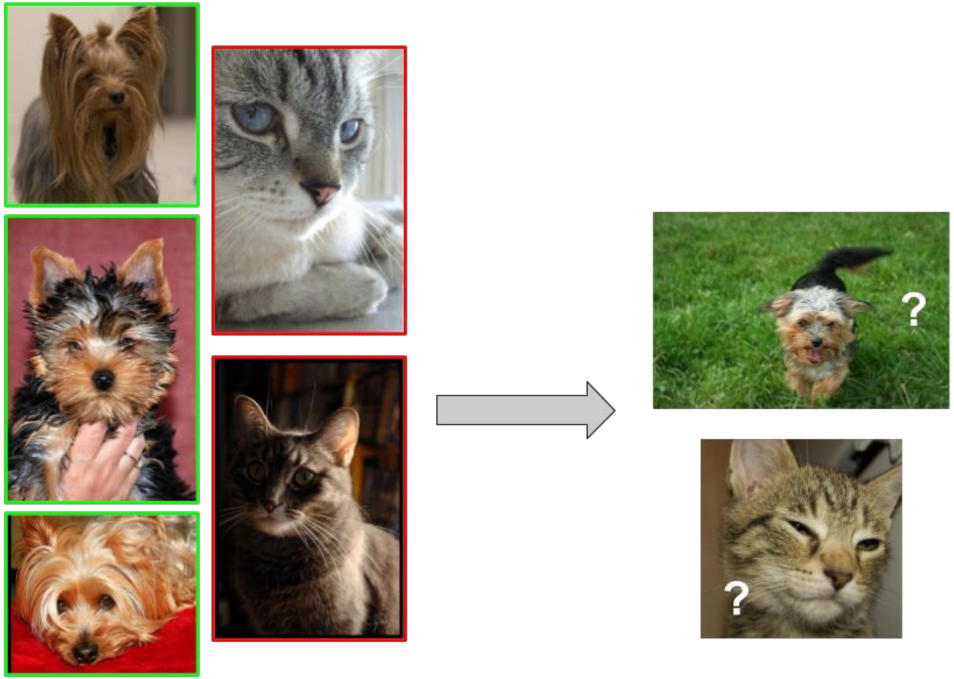
- Introduction & motivation
- Zero-shot learning
  - Definition
  - Side information
  - Zero-shot learning models
  - Exercise
- Low-shot learning
  - Definition
  - Low-shot learning models
- Tips & tricks
- Exercises

Tips & Tricks

Structure

- Introduction & motivation
- Zero-shot learning
    - Definition
    - Side information
    - Zero-shot learning models
    - Exercise
- Low-shot learning
    - Definition
    - Low-shot learning models
- Tips & tricks
- Exercises

Generalization from small training set



http://www.image-net.org/

Overfitting curse

Symptoms

- Very high training accuracy
- Very low testing accuracy

→ Model doesn't generalize to unseen data

Regularization

L_2 regularization

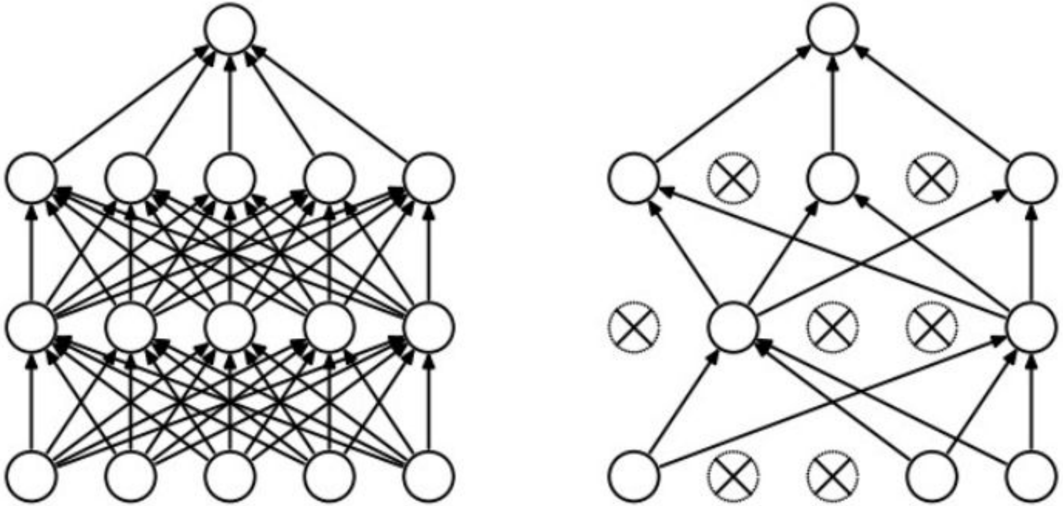
- Most common form of regularization
- Penalizing the squared magnitude of weights in the objective

L_1 regularization

- Relatively common form of regularization
- Penalizing L_1 of weights in the objective

Dropout

Removing a neuron from a designated layer during training or by dropping certain connection



[Srivastava, et al. JMLR2014]
http://cs231n.github.io/neural-networks-2/

Batch normalization

A common practice in NN, forces activiations to have unit gaussian distribution

- Insert BN layer after FC and Conv, and before non-linearities

Robust networks to bad initialization

- interpreted as doing preprocessing at every layer of the network
- Normalization is differentiable

[Ioffe and Christian Szegedy ICML2015]

Transfer learning

Problem Training a model from scratch only with small data is **challenging** and suffer from **overfitting**

Transfer learning

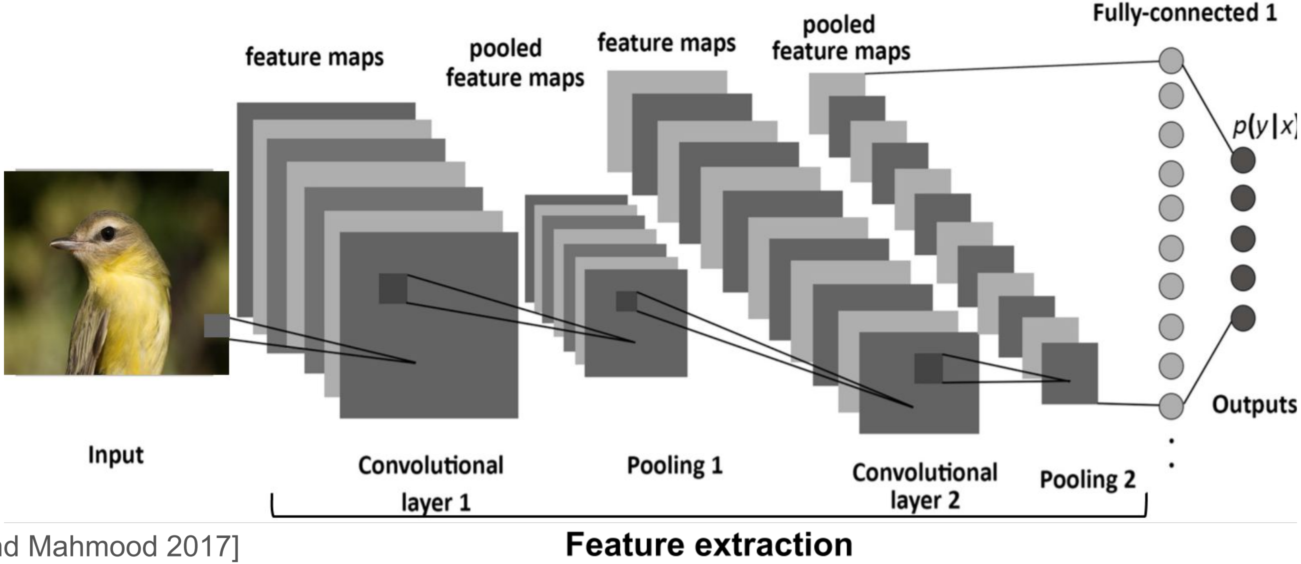
Problem Training a model from scratch only with small data is **challenging** and suffer from **overfitting**

Solution Use the knowledge of a pre-trained model on a border task to solve more specific one

- Bottle neck features
- Fine-tuning top layers

Transfer learning

Extract bottleneck features from a pre-trained network



[Albelwi and Mahmood 2017]

Transfer learning

Finetune top layers of pre-trained network

- Remove top dense layers
- Add your own classification layers on top
- Freeze bottom layers
- Fine-tune top layers on small data

Data augmentation

Increase the amount of training data using information only in our training data

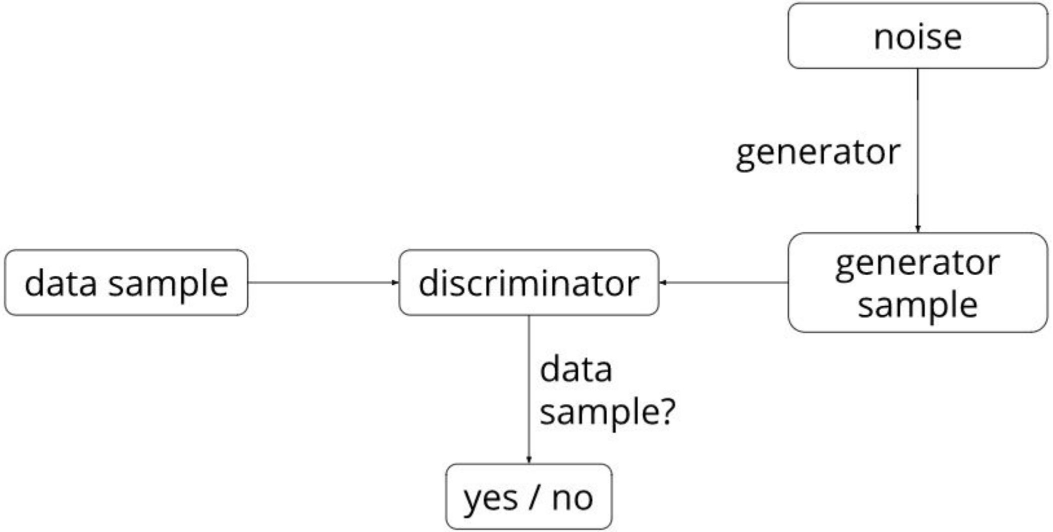
- Affine transformations
- Generative Adversarial Networks (**GANs**)

Affine transformations

https://kids.nationalgeographic.com/animals/meerkat/#meerkat-closeup-log.jpg

Generative Adversarial Networks (GANs)

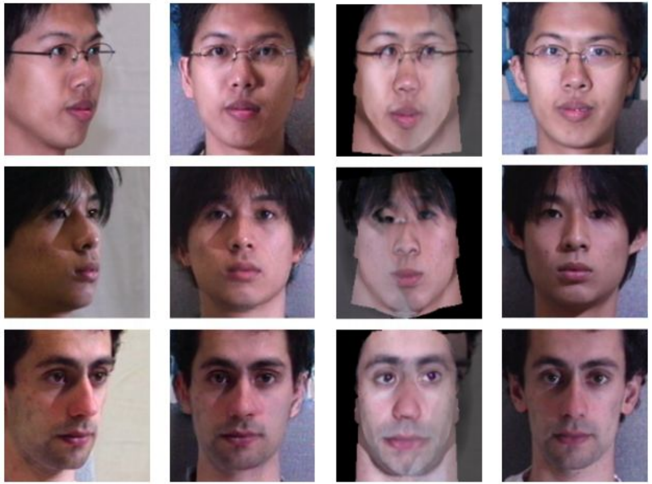
Learns the data distribution



https://ishmaelbelghazi.github.io/ALI/

Generative Adversarial Networks (GANs)

Frontal face generator



GT profile 1 2 GT frontal

[Huang, et al. ICCV2017][1] [Hassner, et al. CVPR2015][2]

Generative Adversarial Networks (GANs)

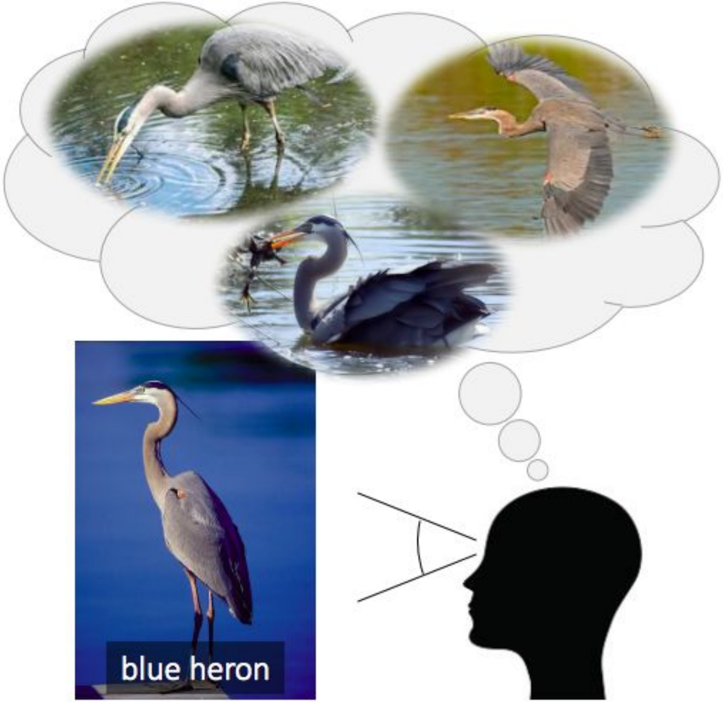
Image to image translation

[Zhu, et al. ICCV2017]
https://github.com/junyanz/CycleGAN

Low-shot learning + GANs

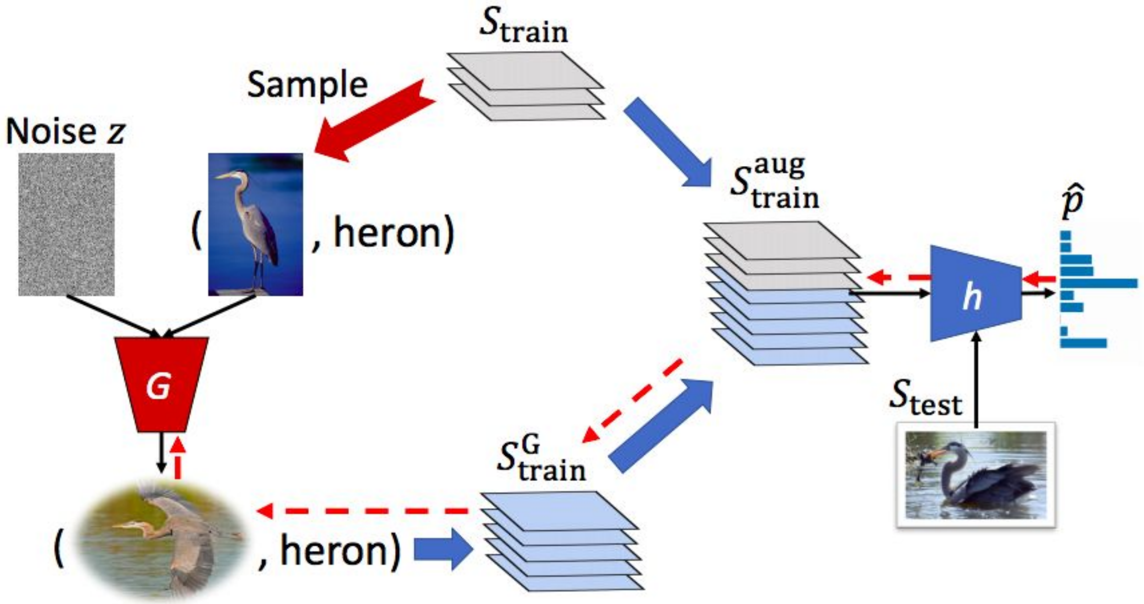
Low-Shot Learning from Imaginary Data

Meta-learning + Hallucination



[Wang, et al. arXiv2018]

Low-shot learning + GANs



[Wang, et al. arXiv2018]

Q & A

Structure

- Introduction & motivation
- Zero-shot learning
  - Definition
  - Side information
  - Zero-shot learning models
  - Exercise
- Low-shot learning
  - Definition
  - Low-shot learning models
- Tips & tricks
- Exercises

Image augmentation exercise

Image augmentation

- Clone git repo
- Use notebook in notebooks/small_classifier/image_augmentation.ipynb
- Read and plot data_aug/meerkat.jpg
- Use keras keras.preprocessing.image.ImageDataGenerator to generate different images with different augmentations
- Plot results

Image classifier with small set exercise

Image classifier with small set

Train small network from scratch

- Clone git repo
- Use notebook in notebooks/small_classifier/image_classifier_with_small_data.ipynb
- Define small conv net: 3 conv blocks (2Dconv,relu_activation,max_pooling) + 2 dense layers (don't forget flatten!)
- Compile network with binary_crossentropy loss and rmsprop
- Define image generator
- Train and validate using generators
- Bonus: plot loss & accuracy

Image classifier with small set

Cheatsheet

Train image 2000
Validation images 800
Input size 150x150x3 (w,h,RGB)
Conv_1: filters 32, kernel size(3,3)
Conv_2: filters 32, kernel size(3,3)
Conv_3: filters 64, kernel size(3,3)
Dense_1: 64, relu
Dense_2: ? , sigmoid

Train small network from scratch

- Clone git repo
- Use notebook in notebooks/small_classifier/image_classifier_with_small_data.ipynb
- Define small conv net: 3 conv blocks (2Dconv,relu_activation,max_pooling) + 2 dense layers (don't forget flatten!)
- Compile network with binary_crossentropy loss and rmsprop
- Define image generator
- Train and validate using generators
- Bonus: plot loss & accuracy

Image classifier with small set

Extract features from pre-trained model

- from keras.applications import vgg16 (set model_top to false)
- Define image generator OR loop through images in data directory
- Use model.predict_generator to get features
- Save features in .npy file

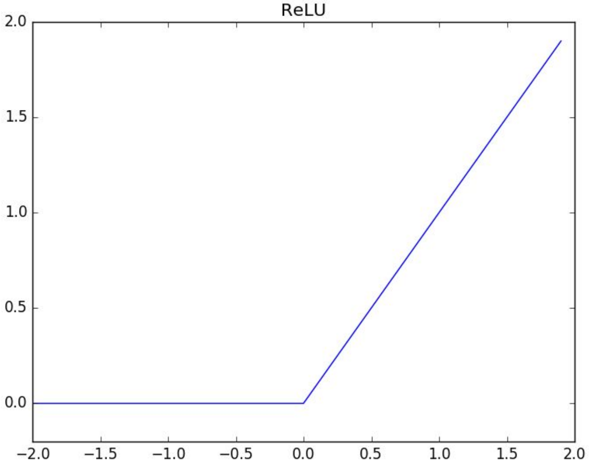
Image classifier with small set

Train small MLP on bottleneck features

- Define network with two dense layers (don’t forget activations and dropout)
- Compile with binary_crossentropy loss and rmsprop
- Train with bottleneck features
- Bonus: plot loss & accuracy

Image classifier with small set

Relu vs. Sigmoid



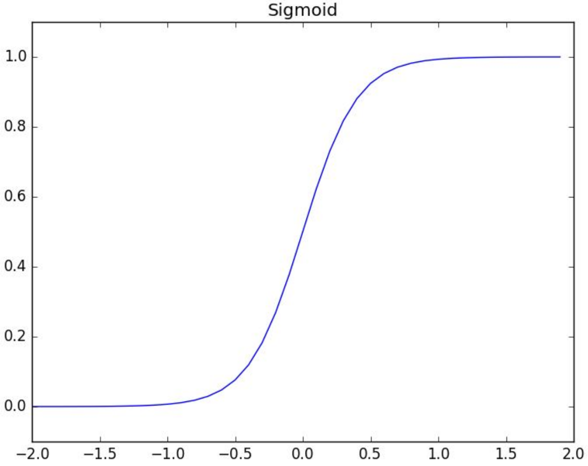
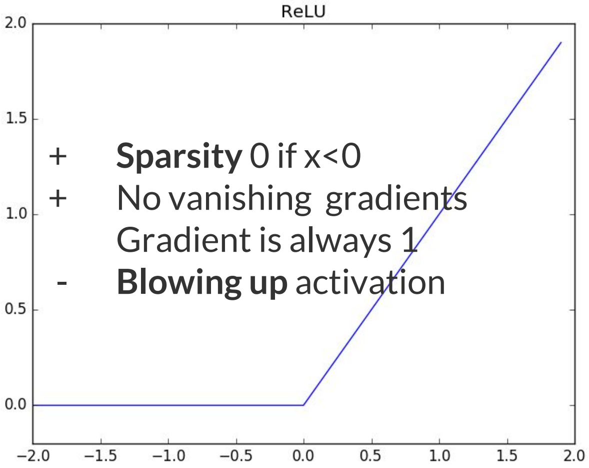
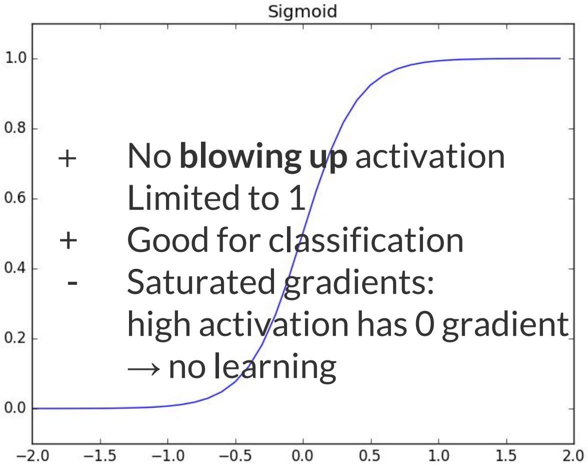


Image classifier with small set

Relu vs. Sigmoid



+ **Sparsity** 0 if x<0
+ No vanishing gradients
  Gradient is always 1
- **Blowing up** activation



+ No **blowing up** activation
  Limited to 1
+ Good for classification
- Saturated gradients:
  high activation has 0 gradient
  → no learning

Image classifier with small set

Cheatsheet

Input size image features size
Flatten 3D feature maps
Dense_1: 256, relu
Dropout: rate 0.5
Dense_2: 1, sigmoid

Train small MLP on bottleneck features

- Define network with two dense layers (don't forget activations and dropout)
- Compile with binary_crossentropy loss and rmsprop
- Train with bottleneck features
- Bonus: plot loss & accuracy

Image classifier with small set

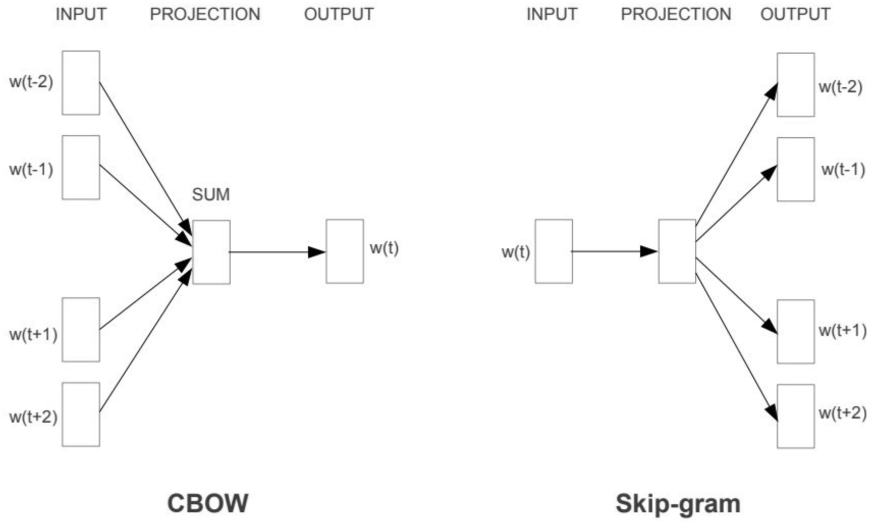
Fine-tune pre-trained network

- Load pre-trained model vgg (weights same as before)
  Note: specify input size according to our images
- Create a new model (vgg + previous mlp)
- Freeze first 15 layers of the new model
- Compile new model with binary_crossentropy loss and SGD with low learning rate (finetuning)
- Train with images

Word Embedding

Word embedding

Dense representations: Word2vec

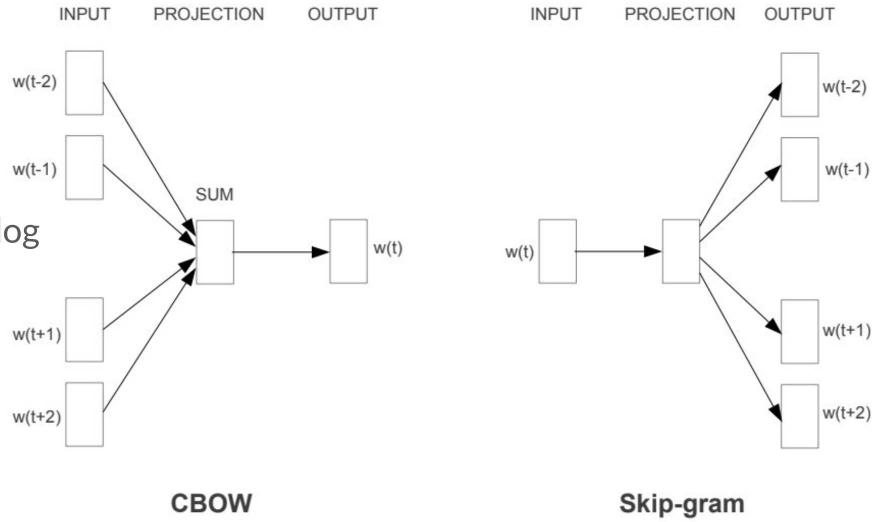


[Mikolov, et al. NIPS2013]

Word embedding

Dense representations: **Word2vec**

The fox __?__ over the lazy dog

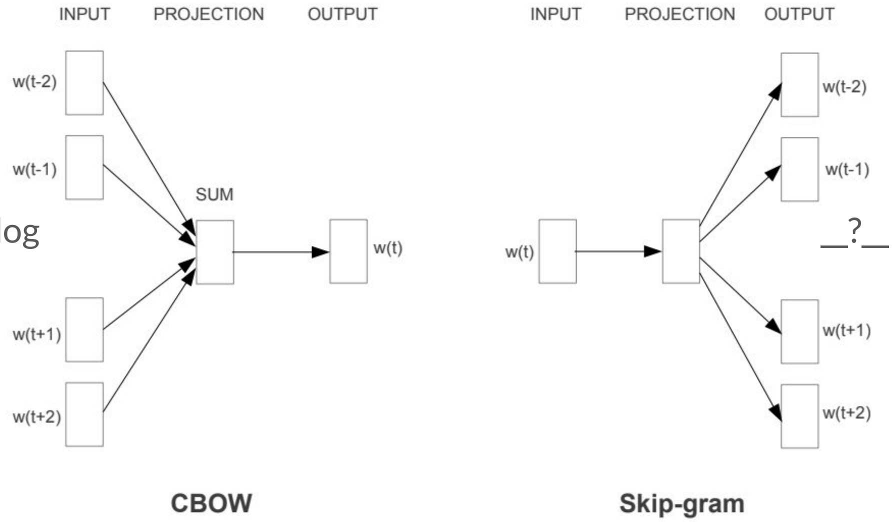


[Mikolov, et al. NIPS2013]

Word embedding

Dense representations: **Word2vec**

The fox __?__ over the lazy dog

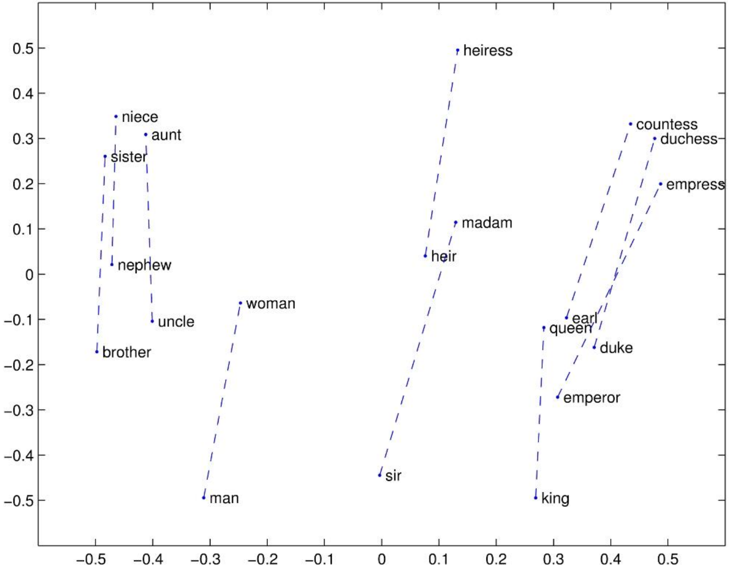


__?__ __?__ jumps __?__ _?_ __?__ __?__

[Mikolov, et al. NIPS2013]

Word embedding

Dense representations: Global vectors for word representation (Glove)



[Pennington, et al. EMNLP2014]

Word embedding exercise

Word embedding exercise

Train your own word2vec

- pip install gensim, tsne, bokeh
- Use sample corpus cloned from repo OR download sample text corpus eng_news_2005_100K here
- Use notebook in notebooks/text_emb/train_word2vec.ipynb
- Train word2vec model using gensim
- Sanity check
- Tsne & plot with bokeh

Word embedding exercise

Load pre-trained Glove

- Download pre-trained model from here
- Use notebook in notebooks/text_emb/Tsne pretrained glove.ipynb
- Load the model using gensim.models.KeyedVectors.load_word2vec_format
  Note: fix first line format
- **Trick**: for quicker loading
- Sanity check
- Tsne & plot with bokeh

Bonus

Interviews QA

- Curse of Dimensionality

Interviews QA

- **Curse of Dimensionality**
  - More features → harder to find a solution

Interviews QA

- **Curse of Dimensionality**
  - More features → harder to find a solution
- **Bias-Variance Tradeoff**

Interviews QA

- **Curse of Dimensionality**
  - More features → harder to find a solution
- **Bias-Variance Tradeoff**
  - Bias: error due to simplistic assumptions in the model

    how well the model fits the data
  - Variance: error due to too much complexity in the model (sensitive for little changes)

    how much the model changes based on changes in the inputs

Interviews QA

- Why Conv layer and not FC for images?

Interviews QA

- **Why Conv layer and not FC for images?**
  - Conv preserves spatial information in the image
  - Conv translation invariant

Interviews QA

- **Why Conv layer and not FC for images?**
  - Conv preserves spatial information in the image
  - Conv translation invariant
- **Max pooling?**

Interviews QA

- **Why Conv layer and not FC for images?**
  - Conv preserves spatial information in the image
  - Conv translation invariant
- **Max pooling?**
  - Reduce computation without losing too much information (max activation)

Interviews QA

- **Why Conv layer and not FC for images?**
    - Conv preserves spatial information in the image
    - Conv translation invariant
- **Max pooling?**
    - Reduce computation without losing too much information (max activation)
- **Normalization?**

Interviews QA

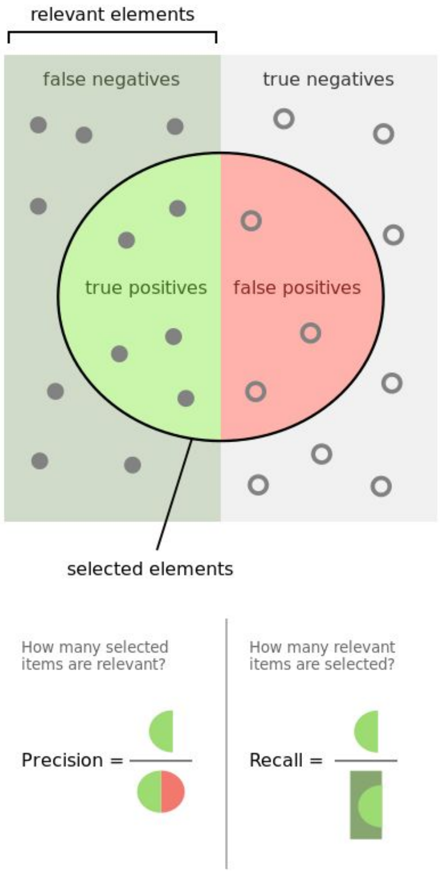
- **Why Conv layer and not FC for images?**
    - Conv preserves spatial information in the image
    - Conv translation invariant
- **Max pooling?**
    - Reduce computation without losing too much information (max activation)
- **Normalization?**
    - makes all features weighted equally → stable convergence

Interviews QA

- Precision vs. Recall

Interviews QA

- **Precision vs. Recall**
  - Recall: amount of positives your model claims compared to the actual number of positives
  - Precision: amount of correct positives your model claims compared to the number of positives it actually claims



https://en.wikipedia.org/wiki/Precision_and_recall

Interviews QA

- F1 score

Interviews QA

- **F1 score**
  - weighted average of the precision and recall of a model
  - 1 the best, 0 the worst.
  - use it in classification where true negatives don't matter much.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Extra resources

Some great resources if you want to dig deeper:

- CVPR tutorial on zero-shot learning
- Matching networks for low-shot learning code implemented with tensorflow and blog post
- Stanford course on convolutional neural networks for visual recognition
- Activation functions comparison
- Blog post about different optimization methods
- Blog post about using embedding layers in neural networks
- Interviews questions springboard, elitedatascience and towardsdatascience

Thanks!