12 - Representation learning

Ludwig Krippahl

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Motivation

Motivation

Features are very important

- Example: long division using Arabic or Roman numerals
- In machine learning, the right representation makes all the difference
- Deep learning can be seen as stacked feature extractors, until the final classification

• The top layer could even be replaced by anothe type of model, in theory

Representation learning

- Supervised learning with limited data can lead to overfitting
- But learning the best representation can be done with unlabelled data

• Unspervised and semi-supervised learning can help find the right features

Motivation

"Meta-priors": what makes a good representation?

Representation Learning; Bengio, Courville, Vincent, 2013

- Manifolds:
  - Actual data is distributed in a subspace of all possible feature value combinations
- Disentanglement:
  - Data is generated by combination of independent factors (e.g. shape, color, lighting, ...)
- Hierarchical organization of explanatory factors:
  - Concepts that explain reality can be composed of more elementar concepts (e.g. edges, shapes, patterns)
- Semi-supervised learning:
  - Unlabelled data is more numerous and can be used to learn structure

"Meta-priors": what makes a good representation?

Representation Learning; Bengio, Courville, Vincent, 2013

- Shared factors:
  - Important features for one problem may also be important for other problems (e.g. image recognition)
- Sparsity:
  - Each example may contain only some of the relevant factors (ears, tail, legs, wings, feet)
- Smoothness:
  - The function we are learning outputs similar y for similar x

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Motivation

"Meta-priors": what makes a good representation?

Representation Learning; Bengio, Courville, Vincent, 2013

- If we can capture these regularities, we can extract useful features from our data
- These features can be reused in different problems, even with different data
- Supervised: the features are learned by the hidden layers to minimize the loss function
- Unsupervised: the same way, but with autoencoders, learning the distribution $P(X)$

Unsupervised pretraining

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Unsupervised pretraining

Greedy layer-wise unsupervised pretraining

- Greedy: optimizes each part independently
- Layer-wise: pretraining is done one layer at a time
  - E.g. train autoencoder, discard decoder, use encoding as input for next layer (another autoencoder)
- Unsupervised: each layer is trained without supervision (e.g. autoencoder)
- Pretraining: the goal is to initialize the network
  - It is followed by fine-tuning with backpropagation
  - Or by training of a classifier "on top" of the pretrained layers

Unsupervised pretraining

Why should this work?

- Initialization has regularizing effect
  - Initially thought as a way to find different local minima, but this does not seem to be the case (ANN do not generally stop at minima)
  - It may be that pretraining allows the network to reach a different region of the parameter space
- Learning the distribution of inputs helps find the right features
  - E.g. unsupervised learning on images identifies salient features (wheels, eyes)
  - These are useful for supervised learning

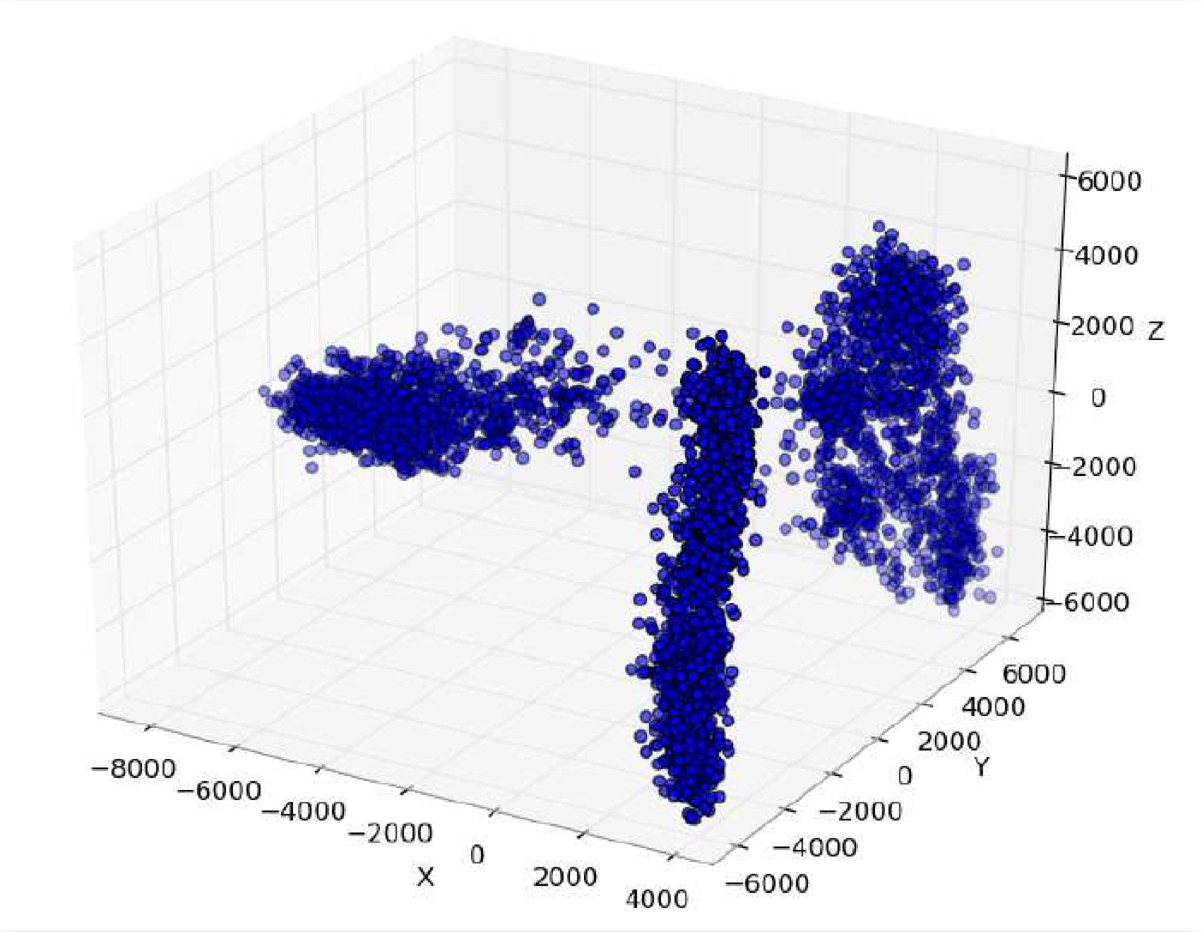
Unsupervised pretraining

When does it work?

- Poor initial representations
  - E.g. word embedding from one-hot vectors
  - One-hot vectors are all equidistant, which is bad for learning
  - Unsupervised pretraining helps find representations that are more useful
- Example: Human actions dataset
  - 5000 dimensions, sparse (around 2% nonzero)
  - Trained denoising autoencoder with L^1 on all data (training, validation, test)
  - PCA on the hidden layer

Unsupervised pretraining

- Poor initial representations



PCA for human actions (5000 sparse binary features) Mesnil et. al, Unsupervised and Transfer Learning JMLR 2011

Unsupervised pretraining

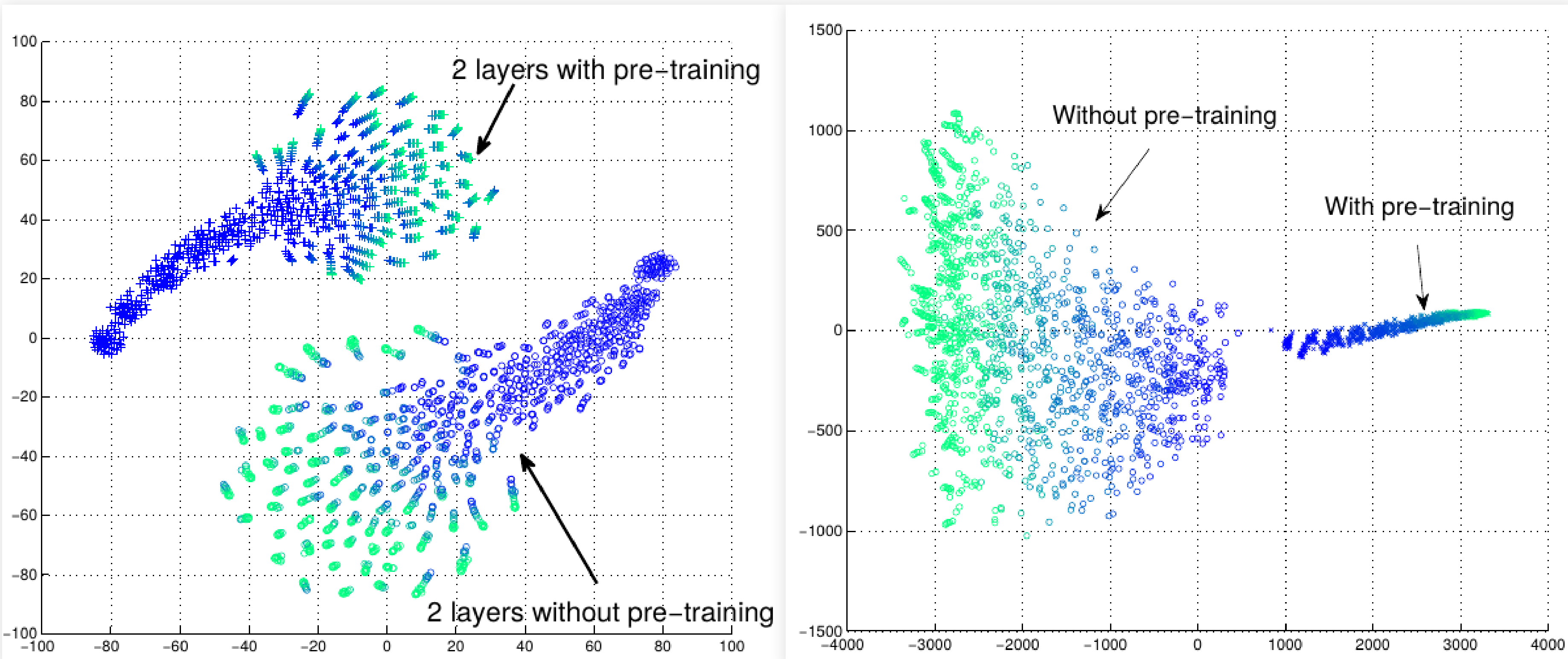
When does it work?

- Poor initial representations
  - E.g. word embedding from one-hot vectors
  - One-hot vectors are all equidistant, which is bad for learning
  - Unsupervised pretraining helps find representations that are more useful
- Regularization, for few labelled examples
  - If labelled data is scarce, there is greater need for regularization and unsupervised pretraining can use unlabelled data for this
- Example: Training trajectories with and without pretraining
  - Concatenate vector of outputs for all test set at different iterations
  - (50 nertworks with and without pretraining)
  - Project into 2D (tSNE and ISOMAP)

Unsupervised pretraining

- Regularizing effect

• Erhan et. al. 2010: output vectors for all data, reduce dimensionality, plot



t-Distributed Stochastic Neighbor Embedding and ISOMAP

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Unsupervised pretraining

Unsupervised pretraining is historically important

- It was the first practical method for training deep networks
  - But has been largely abandoned today because of ReLU and dropout, which allows efficient supervised training and regularization of the whole network
  - For very small datasets, other methods outperform neural networks
  - e.g. Bayesian methods
- Another disadvantage: having two training stages makes it harder to adjust Hyperparameters
- However, still used in some applications, such as natural language processing
  - Unsupervised pretraining with billions of examples to learn good word representations

Transfer learning

Transfer learning

Two different tasks with shared relevant factors

- Shared lower level features:
  - E.g. distinguish between cats and dogs, or between horses and donkeys
  - The low level features are mostly the same, only the higher level classification layers need to change
- Shared higher level representations:
  - E.g. speech recognition
  - The high level generation of sentences is the same for different speakers
  - However, the low level feature extraction may need to be tailored to each speaker

Domain adaptation

Same underlying function but different domains

- We want to model the same mapping from input to output, but are using different sets of examples
- E.g. sentiment analysis
  - Model was trained on customer reviews for movies and songs
  - Now we need to do the same for electronics
  - There should be only one mapping from words to happy or unhappy, but we are training on different sets with different words
  - This is one example where unsupervised training (DAE) can help

Concept drift

Similar to Transfer learning or domain adaptation

- But occurs when the change is gradual over time
- This can be because the actual mapping has changed
  - E.g. changes in the economy change the factors predicting credit risk or purchases
- Or because the data distribution is changing
  - E.g. as the brand becomes more popular, customer base changes from specialized to general

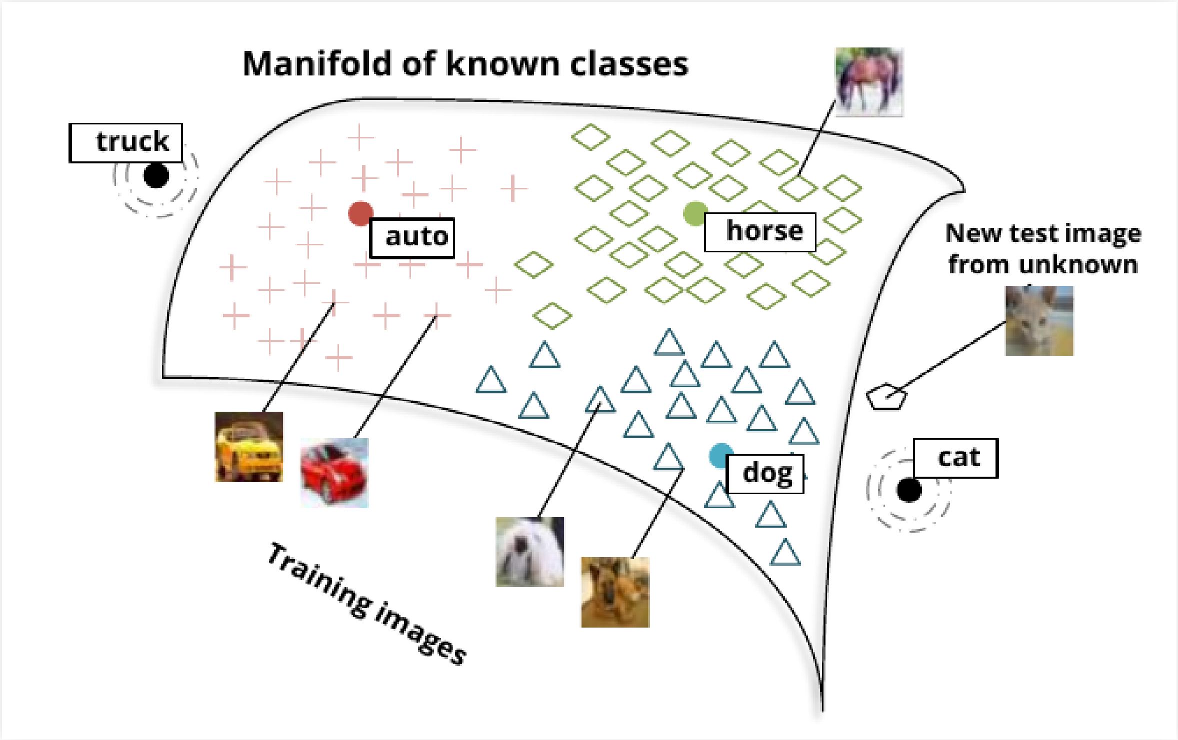
Transfer learning

Use previous experience in new conditions

- The common thread is that we can use what was learned before to help learn now
- Extreme examples: one-shot learning and zero-shot learning
- One-shot learning: use only one labelled example to learn new dataset
  - The rest was learned on other data
- Zero-shot learning: no labelled examples of new classes are necessary
  - Everything was learned on other classes or unsupervised

Transfer learning

- Zero-shot learning, example:
  - Unsupervised learning of word manifold, supervised mapping of known images
  - A new image is mapped to word manifold



Socher et. al. Zero-Shot Learning Through Cross-Modal Transfer (2013)

Supervised pretraining

Take advantage of previously trained models

- E.g. Image recognition networks available in Keras:
  - https://keras.rstudio.com/articles/applications.html

Break down model and problems into simpler parts

- Train one or a few layers at a time, using previously trained as inputs
- Train simpler model in part of the task or set and then add more complexity

Exercise: dimension reduction with autoencoder

Autoencoder

- Compare autoencoder with PCA
- Use the UCI banknote dataset
- Try different architectures
  - e.g. 16, 8, 2, 8, 16 (4)
- Activations and optimizers
  - ReLU, leaky ReLU, Adam, SGD, etc

```
from tensorflow.keras.layers import LeakyReLU
...
layer = Dense(16)(layer)
layer = Activation(LeakyReLU())(layer)
```

- Check learning rates and overfitting

Autoencoder

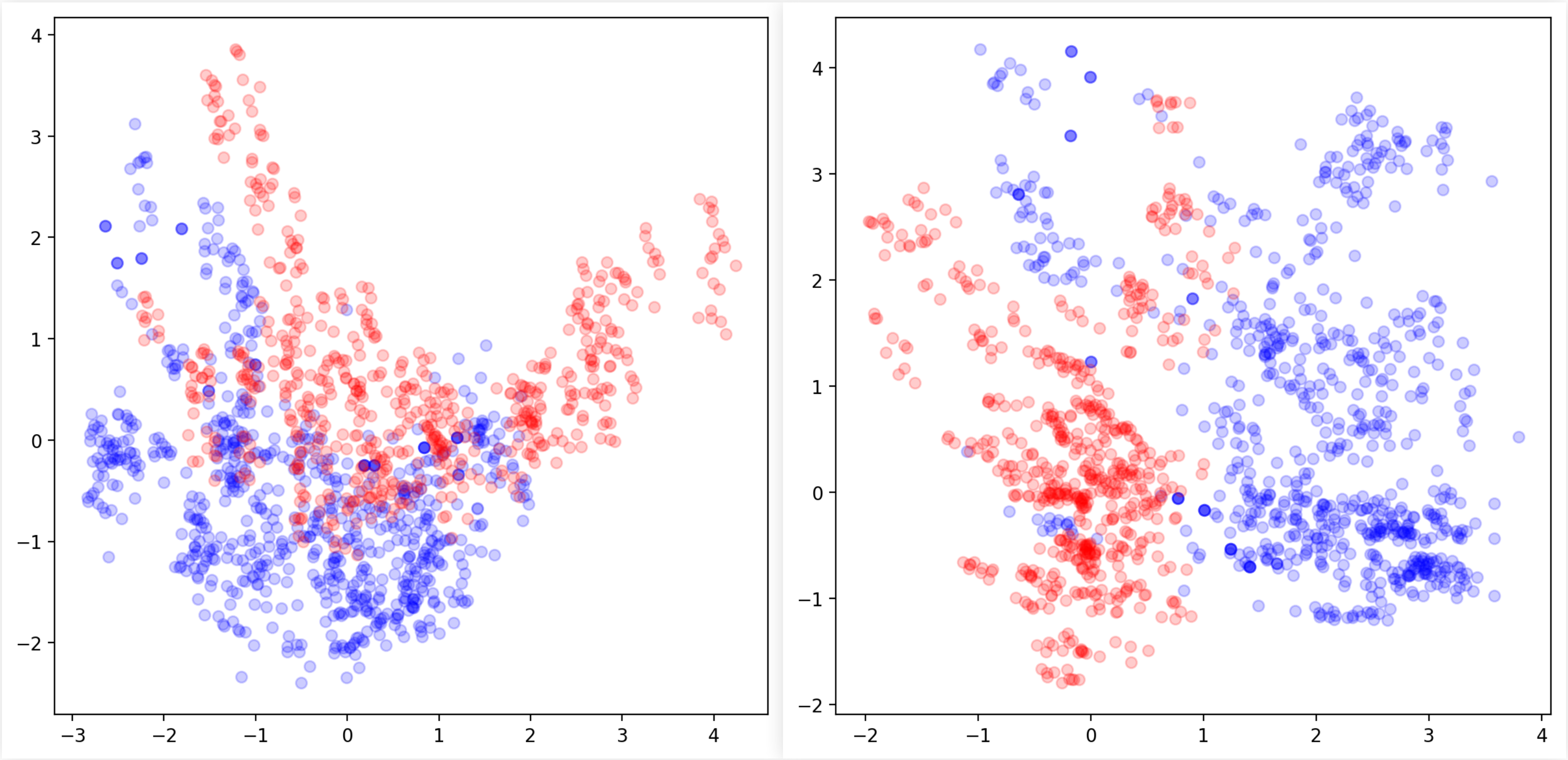
- Compare autoencoder with PCA
- Use the UCI banknote dataset
- After training, get encoded features and compare with PCA

```
from sklearn.decomposition import PCA
...
encoder = Model(inputs = inputs, outputs = model.get_layer('encoded').output)
encoded = encoder.predict(Xs)

pca = PCA(n_components=2)
pca_result = pca.fit_transform(Xs)
```

Autoencoder

- Compare autoencoder with PCA



Summary

Representation learning

Summary

- Improve learning from poor representations
  - Find the best features
- Regularization or feature extraction with unlabelled data
- Historically important in deep learning
  - No longer required but still useful
- Transfer learning (often supervised)

Further reading

- Goodfellow et.al, Deep learning, Chapter 15 (and 8.7.4)
- Bengio et. al. Representation Learning: A Review and New Perspectives, 2013