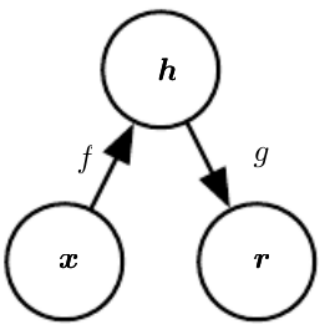
Autoencoders

Victor Kitov

Table of Contents

Definition

- Autoencoder is a neural network, which is trained to reproduce its input.
  - Encoder: $h = f(x)$
  - Decoder: $r = g(h)$



Reconstruction vs. simplicity

- Encoder may learn identity function precisely: $g(f(x)) \equiv x$
  - not useful!
- Need to constrain complexity:
  - by architectural constraint
  - or by penalty on internal representation

  This way encoder needs to learn structure of training set.
- Reconstruction quality vs. representation simplicity tradeoff.
  - extremes: identity or $x_n \to n \to x_n$ vs. constant output.

Details

- Autoencoders were proposed in 1980-s.
- Encoder is a usual feedforward network
  - optimization task

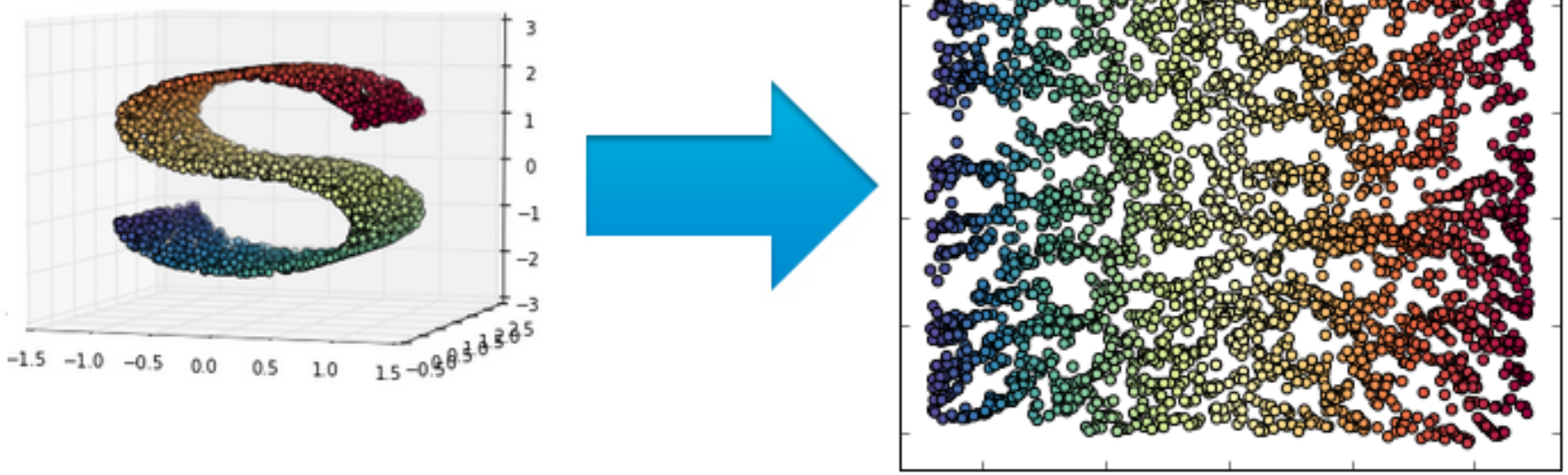
$$\sum_{n=1}^{N} \mathcal{L}(g_\theta(f_\theta(x_n)), x_n) \to \min_\theta$$

  - trained with backpropagation
  - using minibatches

Applications of autoencoders

- Applications:
  - dimensionality reduction
    - visualization
    - feature extraction
    - $\uparrow$ prediction accuracy
    - $\uparrow$ speed of prediction
    - $\downarrow$ memory requirements
  - semantic hashing
  - unsupervised pretraining

Example of dimensionality reduction



Example of manifold

1-D manifold obtained by vertically translating image.

- in space of first 2 principal components
- with tangent line (gray image)

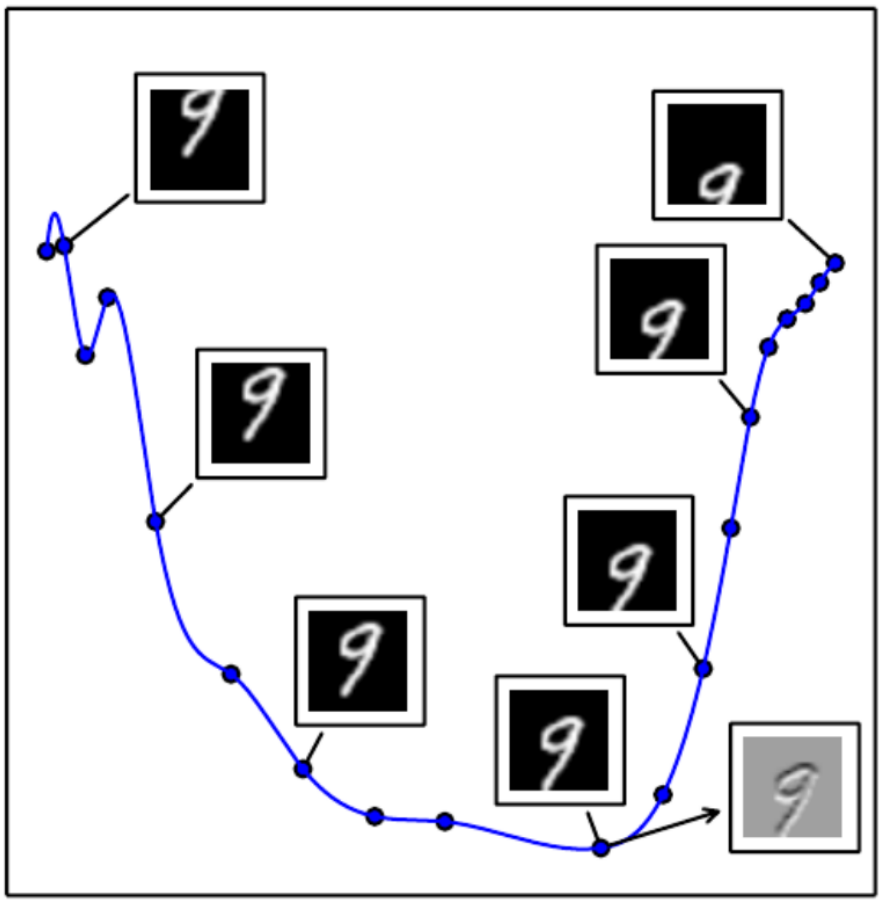
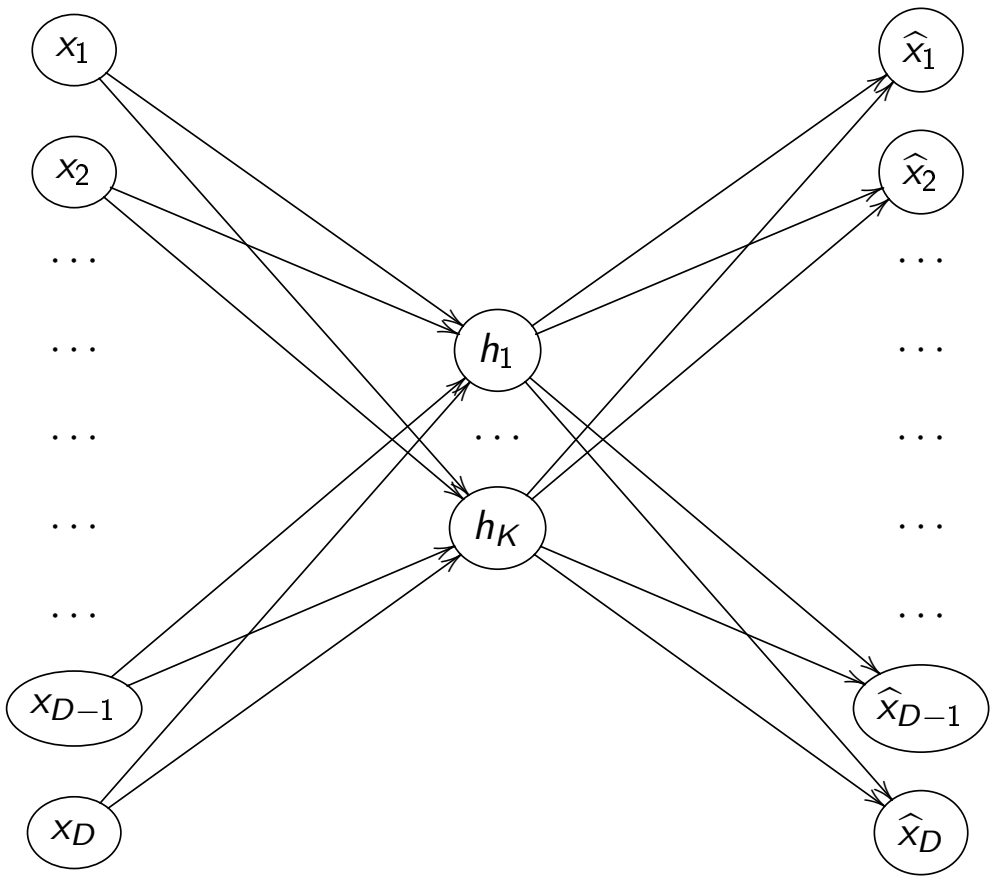


Table of Contents

Undercomplete autoencoders

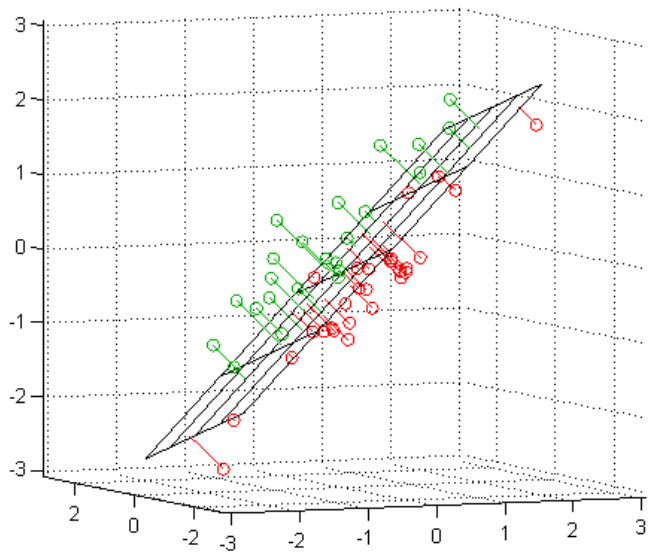
- $x \in \mathbb{R}^D$, $h \in \mathbb{R}^K$
- Autoencoder is undercomplete if $K < D$.



Connection with PCA

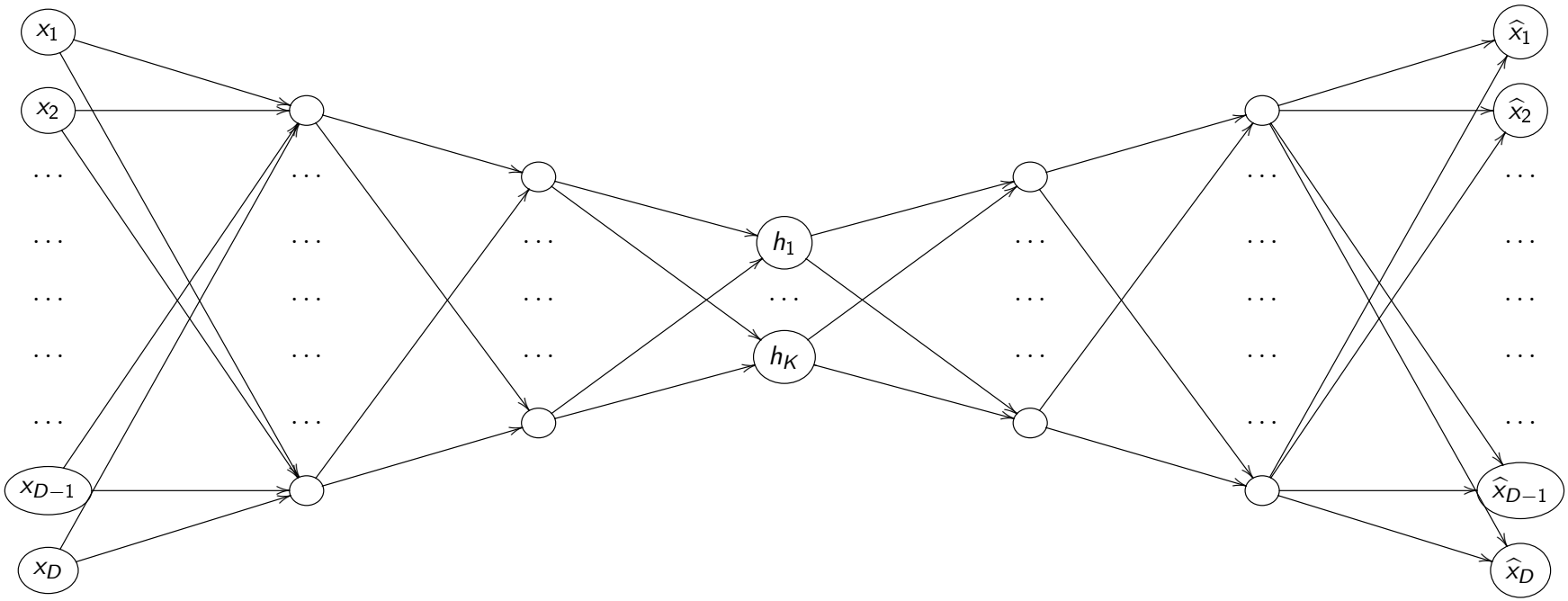
- Undercomplete autoencoder with
  - one hidden layer
  - linear transformations
  - MSE loss $\mathcal{L}(\widehat{x}, x) = \|\widehat{x} - x\|^2$

projects data on subspace of first K principal components.



PCA example

Deep undercomplete autoencoder



Undercomplete autoencoders

- Deep autoencoders may be trained using layerwise unsupervised pretraining.
- Caveat of deep undercomplete autoencoder:
  - capacity is large due to depth
  - $h \in \mathbb{R}$ may encode observation number
  - complicated encoder encodes $x_n \rightarrow n$
  - complicated decoder decodes $n \rightarrow x_n$

2 Autoencoder types

- Undercomplete autoencoders
- Regularized autoencoders
  - Sparse coding
- Denoising autoencoder
- Contractive autoencoders

Regularized autoencoders

- Optimization task

$$\sum_{n=1}^{N} \mathcal{L}(g_\theta(f_\theta(x_n)), x_n) + \lambda R(f_\theta(x_n)) \to \min_\theta$$

- $R(h) = \sum_{d=1}^{D} |h_d|$ induces sparsity.
- To get actual zeros one needs to use linear or ReLu activations.

Sparse coding: definition

- Definitions:
  - $X \in \mathbb{R}^{NxD}$ - design matrix
  - $D \in \mathbb{R}$ - dictionary matrix (rows-code words)
  - $W \in \mathbb{R}$ - representation matrix (rows-object representations)
- Sparse coding is found with optimization task:

$$\|X - WD\|_2^2 + \lambda \|W\|_1 \to \min_{D,W}$$

where $\|A\|_2^2 := \sum_{i,j} a_{i,j}^2$ and $\|A\|_1 := \sum_{i,j} |a_{i,j}|$.

Sparse coding: estimation

- Sparse coding is found with optimization task:

$$\|X - WD\|_2^2 + \lambda \|W\|_1 \to \min_{D,W} \tag{1}$$

- Task (??) is not convex with respect to D, W but is convex with respect to D or W only (holding another matrix fixed).

INPUT: design matrix X

initialize D randomly
while stop condition not met:
    $W = \arg\min_W \|X - WD\|_2^2 + \lambda \|W\|_1$
    $D = \arg\min_D \|X - WD\|_2^2 + \lambda \|W\|_1$

OUTPUT: dictionary D and sparse representation W

- For W solve N LASSO regressions (for each row of W)
- For D solve K OLS regressions (for each column of D)

Sparse coding: encoder & decoder

- Sparse coding is MSE-based sparse autoencoder.
- Suppose we get some observation x
- Encoder $x \to w$
  - dictionary D is fixed
  - solve 1 LASSO regression:

$$\left\|x^T - w^T D\right\|^2 + \lambda \|w\|_1 \to \min_w$$

- Decoder $w \to \widehat{x}$:
  - $\widehat{x}^T = w^T D$

Non-linear analogue

- *Predictive sparse decomposition* (PSD) - generalization of sparse coding autoencoder for non-linear transformations.
- In PSD we minimize:

$$\|x - g_\theta(h)\|^2 + \lambda \|h\|_1 + \gamma \|h - f_\theta(x)\|^2 \to \min_\theta$$

2 Autoencoder types
- Undercomplete autoencoders
- Regularized autoencoders
  - Sparse coding
- Denoising autoencoder
- Contractive autoencoders

Denoising autoencoders

- Solved task:

$$\sum_{n=1}^{N} \mathcal{L}(x, f_\theta(g_\theta(\tilde{x})) \to \min_\theta$$

  where $\tilde{x}$ corresponds to x with added random noise.
- Autoencoder needs to reconstruct structure of the data.
  - recover density $p(x)$
    - to move x away from improbable regions
  - recover typical dependencies between features
    - to reconstruct one feature using other features

Denoising autoencoder

Definition:

1. Sample observation x from training set
2. Sample corrupted version of x with probability $C(\cdot|x)$:

$$\tilde{x} \sim C(\tilde{x}|x)$$

3. Re-estimate parameters using $\mathcal{L}(g_\theta(f_\theta(\tilde{x})) \rightarrow \min_\theta$

Denoising autoencoder

Definition:

1. Sample observation x from training set
2. Sample corrupted version of x with probability $C(\cdot|x)$:

$$\tilde{x} \sim C(\tilde{x}|x)$$

3. Re-estimate parameters using $\mathcal{L}(g_\theta(f_\theta(\tilde{x})) \rightarrow \min_\theta$

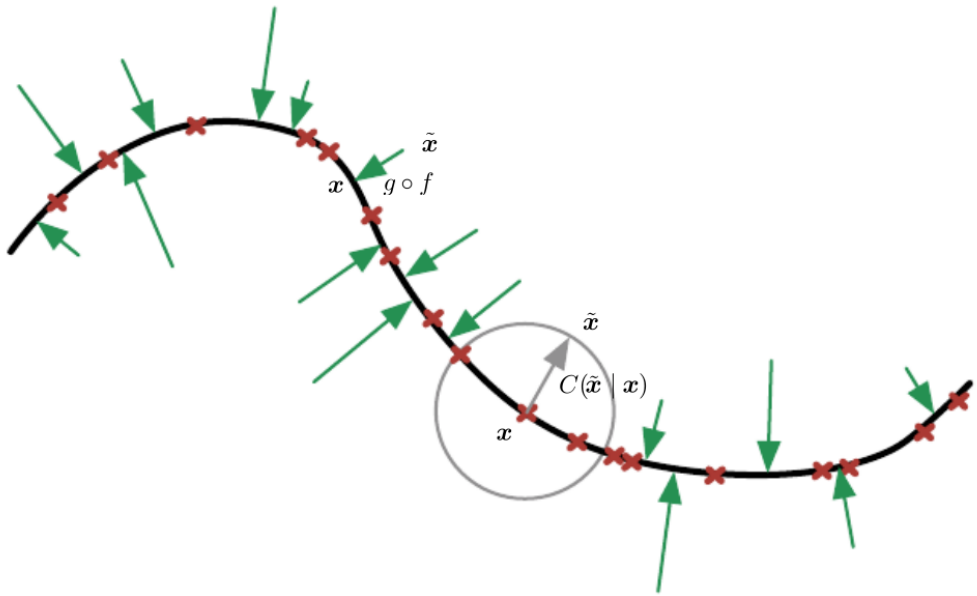
Common types of noise:

- Gaussian $N(0, \sigma^2 I)$, $I \in \mathbb{R}^D$
- Binary mask:
  - for each $d = 1, 2, ...D$ with probability p: $x^d \leftarrow 0$.

Benefits

- Autoencoder does not learn identity
  - otherwise it will propagate noise
- Noise forces autoencoder to learn good model representation of data
  - Density distribution
  - Dependencies between features

What it learns



- Gray circle: corruption area
- Black line: manifold, where objects are concentrated.
- Red crosses: training set.
- Green lines: $g(f(x)) - x$
  - manifold learning! manifold points satisfy $g(f(x)) = x$ (necessary condition) and manifold directions Δx: $\|\nabla_x \{f(x))\} \Delta x\|$ - large

2 Autoencoder types

- Undercomplete autoencoders
- Regularized autoencoders
  - Sparse coding
- Denoising autoencoder
- Contractive autoencoders

Contractive autoencoders

- Solved task:

$$\sum_{n=1}^{N}\left\{\mathcal{L}(x, f_\theta(g_\theta(x_n)) + \lambda \left\|J_f(x_n)\right\|_F^2\right\} \to \min_\theta$$

  where $\left\|J_f(x)\right\|_F^2 = \sum_{k=1}^{K}\sum_{d=1}^{D}\left(\frac{\partial f_k(x)}{\partial x^d}\right)^2$
- Jacobian reduces sensitivity of h to x
- Such regularization $\uparrow$robustness of representation to small variations in x.
- λ controls tradeoff between reconstruction error and robustness.

Contractive autoencoders and manifolds

- $h = f(x)$ maps input to coordinates in embedded space.
- $f(x + \Delta x) \approx f(x) + J_f(x)\Delta x$
- Directions Δx:
  - with large $\|J_f(x)\Delta x\|$ are tangent to manifold
  - with small $\|J_f(x)\Delta x\|$ are perpendicular to manifold

Contractive vs. denoising autoencoders[1]

Denoising autoencoder becomes equivalent to contractive autoencoder under 2 conditions:

- denoising autoencoder: for infinitesimal Gaussian noise
- contractive autoencoder: for penalty on reconstruction $r(x)$ rather than on $f(x)$.

[1] See Alain and Bengio (2013).

Caveat

Trivial solution to contractive autoencoder:

$$\begin{cases} f(x) \to \varepsilon f(x) \\ g(h) \to \frac{1}{\varepsilon} g(h) \end{cases}$$

Possible way-out:

- use tied weights in encoder and decoder.

Empirical results

Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. ICML. Rifai et al. 2011.

	Model	Test error	Average $\|J_f(x)\|_F$
MNIST	CAE	**1.14**	$0.73\ 10^{-4}$
	DAE-g	**1.18**	$0.86\ 10^{-4}$
	RBM-binary	1.30	$2.50\ 10^{-4}$
	DAE-b	1.57	$7.87\ 10^{-4}$
	AE+wd	1.68	$5.00\ 10^{-4}$
	AE	1.78	$17.5\ 10^{-4}$
CIFAR-bw	CAE	**47.86**	$2.40\ 10^{-5}$
	DAE-b	**49.03**	$4.85\ 10^{-5}$
	DAE-g	54.81	$4.94\ 10^{-5}$
	AE+wd	55.03	$34.9\ 10^{-5}$
	AE	55.47	$44.9\ 10^{-5}$

Empirical results: details

- One hidden layer in classification network
- Sigmoidal units, cross-entropy score
- Unsupervised pretraining with each autoencoder
- Definitions:
  - DAE - denoising autoencoder
  - b-binary mask noise
  - wd - weight decay
  - AE - undercomplete autoencoder
  - g-Gaussian

Table of Contents

Depth of autoencoders

- To obtain h with desired properties we need at least 1 hidden layer before central layer (with output h)
  - according to universal approximator theorem
- Depth can exponentially reduce number of neurons for representing some functions.
- Deep autoencoders give better data compression
- Deep autoencoder is usually trained with *greedy layerwise pretraining*.

Stochastic autoencoders

Usual autoencoder:

- $x \to h$
- $h \to x$

Stochastic encoder:

- $x \to p_{encoder}(h|x)$

Stochastic decoder:

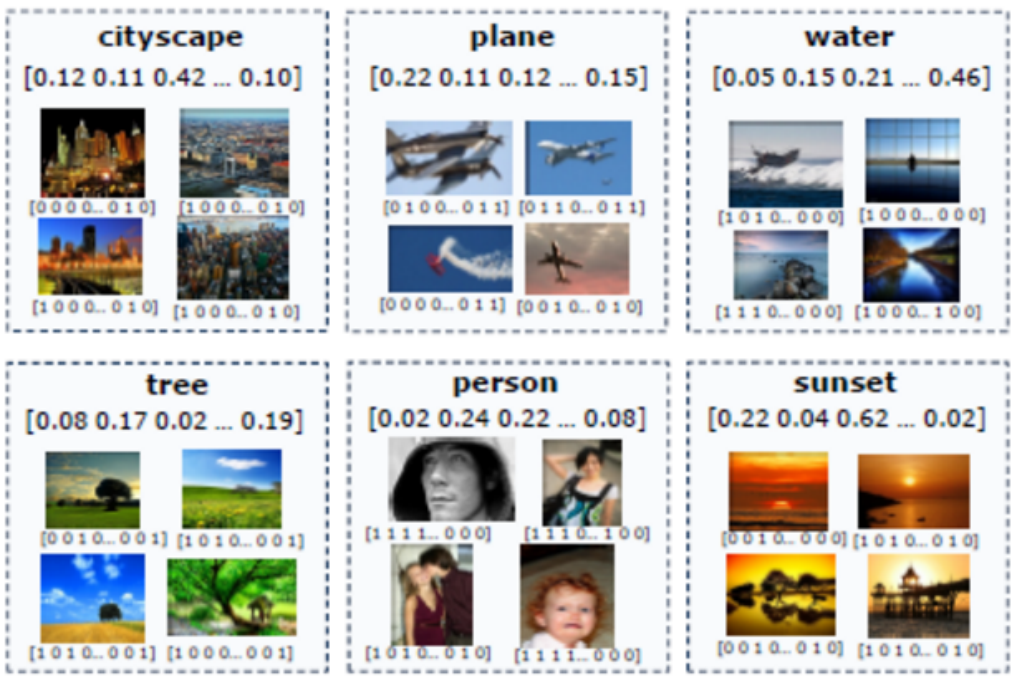
- $h \to p_{decoder}(x|h)$

Stochastic autoencoder may be estimated with maximum likelihood.

Table of Contents

Semantic hashing

- Map complicated objects (e.g. texts, images) to binary codes.
- Objects with the same binary code are similar
  - may also consider objects which have almost the same binary code
    - by flipping several bits



Binarization of representation

- To make binary codes:
  - use sigmoid non-linearity
  - before this non-linearity add noise
    - to confront this noise model will need to make activations very large or small - sigmoid will saturate in both cases.

Table of Contents

- Sparse coding

5. Pretraining
   - Unsupervized pretraining
   - Supervised pretraining

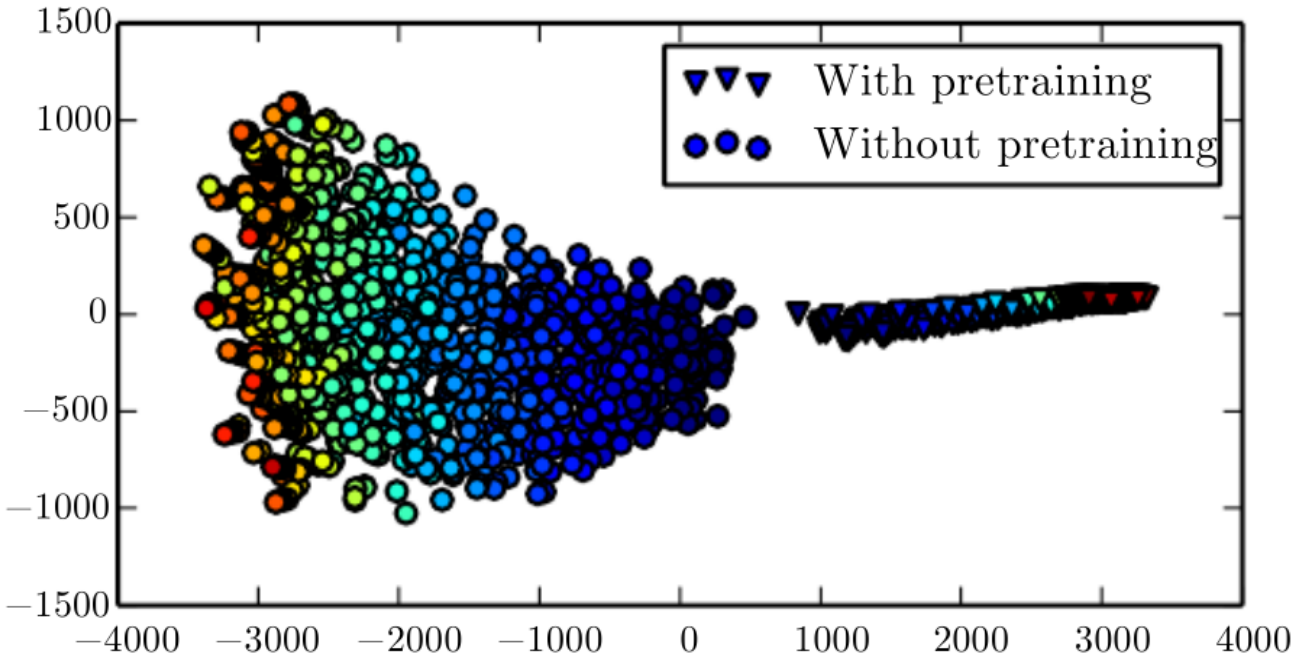
Unsupervized pretraining

- Greedy Layer-Wise Unsupervised Pretraining:
  1. By training shallow autoencoders build layer by layer
  2. Use pretrained layers as initialization for deep network

Unsupervized pretraining

- Greedy Layer-Wise Unsupervised Pretraining:
  1. By training shallow autoencoders build layer by layer
  2. Use pretrained layers as initialization for deep network
- Work often better
  - learns internal data representation
    - may be useful features
  - initializes optimization from more favorable initial approximation
    - good for solving vanishing gradient problem
    - we may fix pretrained weights: e.g. word2vec.
  - especially useful when few labelled examples and many unlabelled

Demo[2]



- Model predictions were transformed to 2D using Isomap (tries to preserve global distances).
- Color indicates epoch of neural net training.
- Many random initializations were considered.

[2] "Why Does Unsupervised Pre-training Help Deep Learning?" Erhan et. al. JLMR. 2010.

Experiment

Deep Sparse Rectifier Neural Networks. Glorot et. al. 2011. JMLR.

Table: **Test error on networks of depth 3.** Bold results represent statistical equivalence between similar experiments, with and without pre-training, under the null hypothesis of the pairwise test with $p = 0.05$.

Neuron	MNIST	CIFAR10	NISTP	NORB
***With* unsupervised pre-training**				
Rectifier	**1.20%**	**49.96%**	**32.86%**	**16.46%**
Tanh	**1.16%**	**50.79%**	35.89%	17.66%
Softplus	**1.17%**	**49.52%**	**33.27%**	19.19%
***Without* unsupervised pre-training**				
Rectifier	**1.43%**	**50.86%**	**32.64%**	**16.40%**
Tanh	1.57%	52.62%	36.46%	19.29%
Softplus	1.77%	53.20%	35.48%	17.68%

- Details:
  - 3 hidden layers stacked denoising autoencoder
  - noise: binary mask with 0.25 probability of zero.
- Caveats:
  - networks of depth 3
  - no dropout & batch normalization used.

Drawbacks

- No direct connection between features well suited for classification and discrimination.
  - some papers indicate that pretraining may harm solution
- How to select hyper-parameters of pretrainer?
  - using CV on final supervised task but this is long
  - must fallback to unsupervised quality criteria
- Either random initialization or unsupervised pretraining - no intermediate solution.
  - Possible way-out[3]:

$$\mathcal{L}_{hibrid}(D_{train}) = \mathcal{L}_{discriminative}(D_{train}) + \alpha \mathcal{L}_{generative}(D_{train})$$

- Drop-out, batch-norm and ReLu give possibility to achieve good results without unsupervised pretraining.

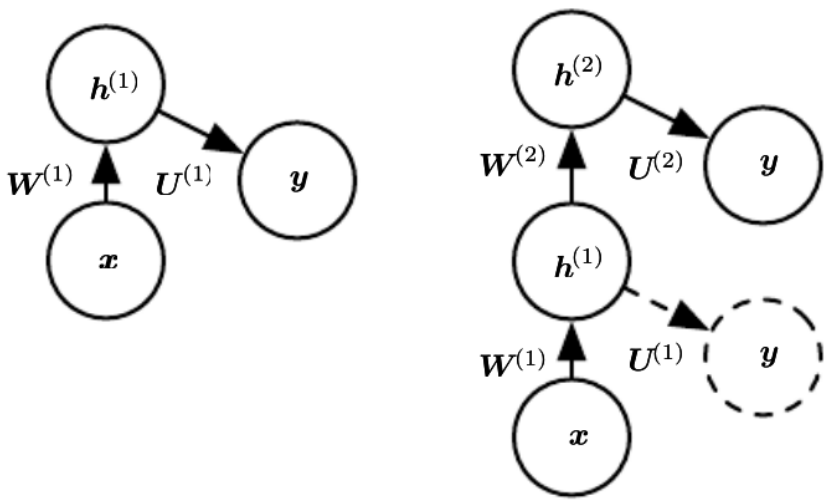
[3] See "Classification using Discriminative Restricted Boltzmann Machines". Larochelle and Bengio ICML-2008.

- Sparse coding

5. Pretraining
   - Unsupervized pretraining
   - Supervised pretraining

Supervised pretraining

- Supervised pretraining:
  - Learn layer, fix it, learn next layer, etc.



Details

- Fine-tuning of whole network
  - We may refit all previous layers:
    - after each step of pretraining
    - or once after all layers were pretrained
- Frequently used strategy in transfer learning
  - example: in images, e.g. AlexNet pretrained weights.