CSCE 496/896 Lecture 5: Autoencoders

Stephen Scott

(Adapted from Paul Quint and Ian Goodfellow)

sscott@cse.unl.edu

CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Introduction

- **Autoencoding** is training a network to replicate its input to its output
- Applications:
  - Unlabeled pre-training for semi-supervised learning
  - Learning **embeddings** to support information retrieval
  - Generation of new instances similar to those in the training set
  - Data compression

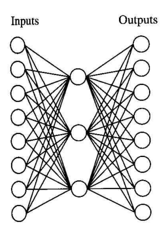
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Outline

- Basic idea
- Stacking
- Types of autoencoders
  - Denoising
  - Sparse
  - Contractive
  - Variational
  - Generative adversarial networks

CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Basic Idea



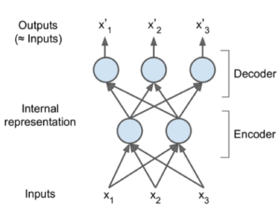
Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

- Sigmoid activation functions, 5000 training epochs, square loss, no regularization
- What's special about the hidden layer outputs?

CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Basic Idea

- An **autoencoder** is a network trained to learn the **identity function:** output = input



- Subnetwork called **encoder** $f(\cdot)$ maps input to an **embedded representation**
- Subnetwork called **decoder** $g(\cdot)$ maps back to input space

- Can be thought of as **lossy compression** of input
- Need to identify the important attributes of inputs to reproduce faithfully

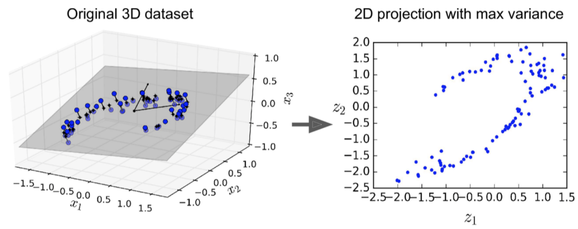
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Basic Idea

- General types of autoencoders based on size of hidden layer
  - **Undercomplete** autoencoders have hidden layer size smaller than input layer size
    - ⇒ Dimension of embedded space lower than that of input space
    - ⇒ Cannot simply memorize training instances
  - **Overcomplete** autoencoders have much larger hidden layer sizes
    - ⇒ Regularize to avoid overfitting, e.g., enforce a **sparsity** constraint

Basic Idea

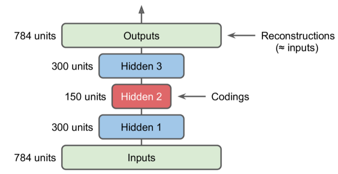
Example: Principal Component Analysis



- A 3-2-3 autoencoder with linear units and square loss performs **principal component analysis**: Find linear transformation of data to maximize variance

Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Stacked Autoencoders



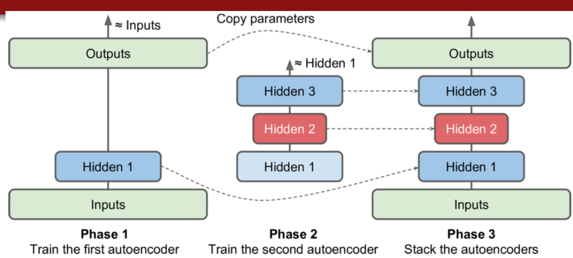
- A **stacked autoencoder** has multiple hidden layers
- Can share parameters to reduce their number by exploiting symmetry: $W_4 = W_1^\top$ and $W_3 = W_2^\top$

```
weights1 = tf.Variable(weights1_init, dtype=tf.float32, name="weights1")
weights2 = tf.Variable(weights2_init, dtype=tf.float32, name="weights2")
weights3 = tf.transpose(weights2, name="weights3")      # shared weights
weights4 = tf.transpose(weights1, name="weights4")      # shared weights
```


Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Stacked Autoencoders

Incremental Training

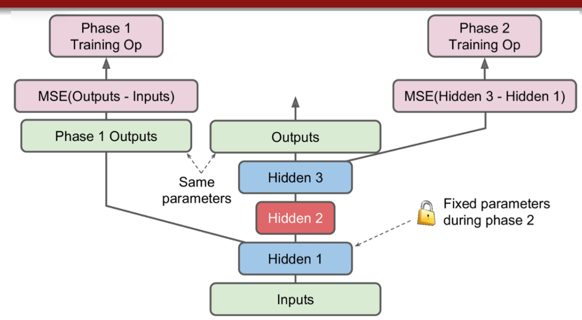


- Can simplify training by starting with single hidden layer H_1
- Then, train a second AE to mimic the output of H_1
- Insert this into first network
- Can build by using H_1's output as training set for Phase 2

Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Stacked Autoencoders

Incremental Training (Single TF Graph)



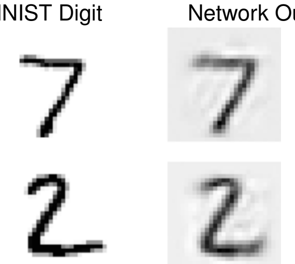
- Previous approach requires multiple TensorFlow graphs
- Can instead train both phases in a single graph: First left side, then right

Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Stacked Autoencoders

Visualization

Input MNIST Digit   Network Output

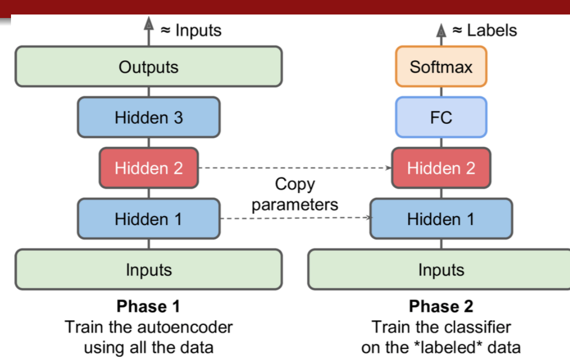


Weights (features selected) for five nodes from H_1:

Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Stacked Autoencoders

Semi-Supervised Learning

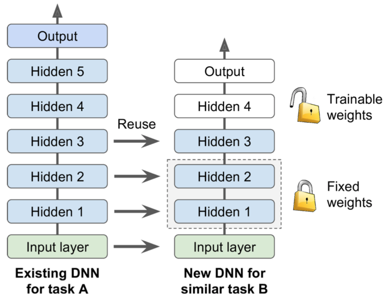


- Can **pre-train** network with unlabeled data

⇒ learn useful features and then train "logic" of dense layer with labeled data

Transfer Learning from Trained Classifier

- Can also transfer from a classifier trained on different task, e.g., transfer a GoogleNet architecture to ultrasound classification



- Often choose existing one from a **model zoo**

Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

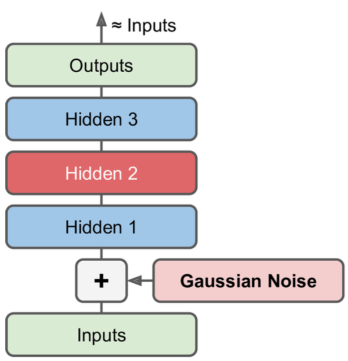
Denoising Autoencoders

Vincent et al. (2010)

- Can train an autoencoder to learn to **denoise** input by giving input **corrupted** instance $\tilde{x}$ and targeting **uncorrupted** instance x
- Example noise models:
  - **Gaussian noise:** $\tilde{x} = x + z$, where $z \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$
  - **Masking noise:** zero out some fraction ν of components of x
  - **Salt-and-pepper noise:** choose some fraction ν of components of x and set each to its min or max value (equally likely)

Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

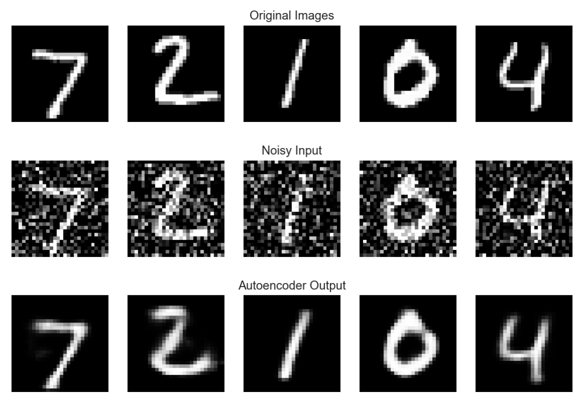
Denoising Autoencoders



Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Denoising Autoencoders

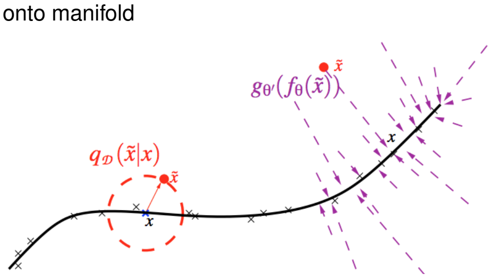
Example



Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Denoising Autoencoders

- How does it work?
- Even though, e.g., MNIST data are in a 784-dimensional space, they lie on a low-dimensional **manifold** that captures their most important features
- Corruption process moves instance x off of manifold
- Encoder f_θ and decoder $g_{\theta'}$ are trained to project $\tilde{x}$ back onto manifold



Nebraska Lincoln
CSCE 496/896 Lecture 5: Autoencoders
Stephen Scott
Introduction
Basic Idea
Stacked AE
Denoising AE
Sparse AE
Contractive AE
Variational AE
GAN

Sparse Autoencoders

- An overcomplete architecture
- Regularize outputs of hidden layer to enforce **sparsity**:

$$\tilde{\mathcal{J}}(\boldsymbol{x}) = \mathcal{J}(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \alpha\,\Omega(\boldsymbol{h}) \ ,$$

  where $\mathcal{J}$ is loss function, f is encoder, g is decoder, $\boldsymbol{h} = f(\boldsymbol{x})$, and Ω penalizes non-sparsity of $\boldsymbol{h}$
- E.g., can use $\Omega(\boldsymbol{h}) = \sum_i |h_i|$ and ReLU activation to force many zero outputs in hidden layer
- Can also measure average activation of h_i across mini-batch and compare it to user-specified **target sparsity** value p (e.g., 0.1) via square error or **Kullback-Leibler divergence**:

$$p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q} \ ,$$

  where q is average activation of h_i over mini-batch

Contractive Autoencoders

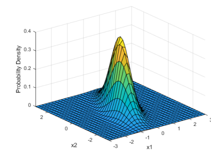
- Similar to sparse autoencoder, but use

$$\Omega(\boldsymbol{h}) = \sum_{j=1}^{m} \sum_{i=1}^{n} \left(\frac{\partial h_i}{\partial x_j} \right)^2$$

- I.e., penalize large partial derivatives of encoder outputs wrt input values
- This **contracts** the output space by mapping input points in a neighborhood near $\boldsymbol{x}$ to a smaller output neighborhood near $f(\boldsymbol{x})$
  - $\Rightarrow$ Resists perturbations of input $\boldsymbol{x}$
- If $\boldsymbol{h}$ has sigmoid activation, encoding near binary and a CE pushes embeddings to corners of a hypercube

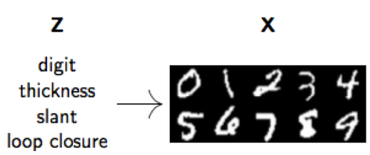
Variational Autoencoders

- VAE is an autoencoder that is also **generative model**
  - $\Rightarrow$ Can generate new instances according to a probability distribution
  - E.g., hidden Markov models, Bayesian networks
  - Contrast with **discriminative models**, which predict classifications
- Encoder f outputs $[\boldsymbol{\mu}, \boldsymbol{\sigma}]^\top$
  - Pair (μ_i, σ_i) parameterizes Gaussian distribution for dimension $i = 1, \ldots, n$
  - Draw $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$
  - Decode this **latent variable** z to get $g(z)$

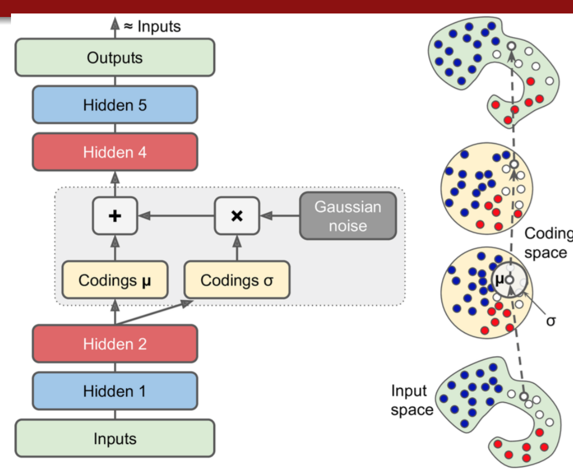


Variational Autoencoders
Latent Variables

- Independence of z dimensions makes it easy to generate instances wrt complex distributions via decoder g
- Latent variables can be thought of as values of attributes describing inputs
  - E.g., for MNIST, latent variables might represent "thickness", "slant", "loop closure"



Variational Autoencoders
Architecture



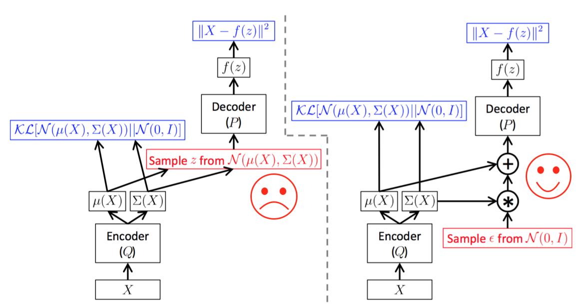
Variational Autoencoders
Optimization

- **Maximum likelihood** (ML) approach for training generative models: find a model ($\boldsymbol{\theta}$) with maximum probability of generating the training set $\mathcal{X}$
- Achieve this by minimizing the sum of:
  - End-to-end AE loss (e.g., square, cross-entropy)
  - Regularizer measuring distance (K-L divergence) from latent distribution $q(z \mid \boldsymbol{x})$ and $\mathcal{N}(\boldsymbol{0}, I)$ (= standard multivariate Gaussian)
- $\mathcal{N}(\boldsymbol{0}, I)$ also considered the **prior distribution** over z (= distribution when no $\boldsymbol{x}$ is known)

```
eps = 1e-10
latent_loss = 0.5 * tf.reduce_sum(
        tf.square(hidden3_sigma) + tf.square(hidden3_mean)
        - 1 - tf.log(eps + tf.square(hidden3_sigma)))
```


Variational Autoencoders
Reparameterization Trick

- Cannot backprop error signal through random samples
- **Reparameterization trick** emulates $z \sim \mathcal{N}(\mu, \sigma)$ with $\epsilon \sim \mathcal{N}(0, 1)$, $z = \epsilon\sigma + \mu$

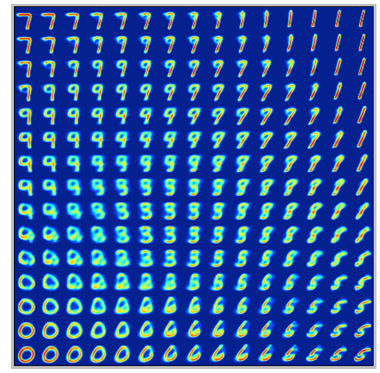


Variational Autoencoders
Example Generated Images: Random

- Draw $z \sim \mathcal{N}(\mathbf{0}, I)$ and display $g(z)$

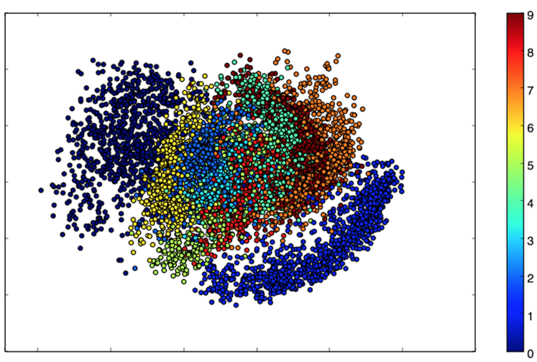
Variational Autoencoders
Example Generated Images: Manifold

- Uniformly sample points in z space and decode



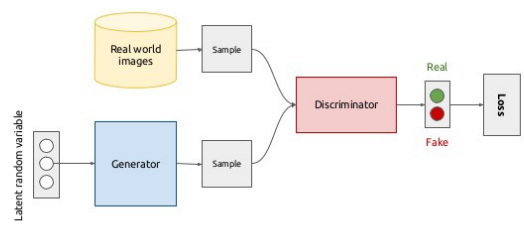
Variational Autoencoders
2D Cluster Analysis

- Cluster analysis by digit



Generative Adversarial Network

- GANs are also generative models, like VAEs
- Models a game between two players
  - **Generator** creates samples intended to come from training distribution
  - **Discriminator** attempts to discern the "real" (original training) samples from the "fake" (generated) ones
- Discriminator trains as a binary classifier, generator trains to fool the discriminator



Generative Adversarial Network
How the Game Works

- Let $D(\boldsymbol{x})$ be discriminator parameterized by $\boldsymbol{\theta}^{(D)}$
  - Goal: Find $\boldsymbol{\theta}^{(D)}$ minimizing $J^{(D)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$
- Let $G(\boldsymbol{z})$ be generator parameterized by $\boldsymbol{\theta}^{(G)}$
  - Goal: Find $\boldsymbol{\theta}^{(G)}$ minimizing $J^{(G)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$
- A **Nash equilibrium** of this game is $\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$ such that each $\boldsymbol{\theta}^{(i)}$, $i \in \{D, G\}$ yields a local minimum of its corresponding J

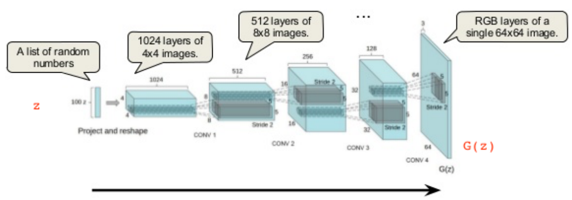
Generative Adversarial Network
Training

- Each training step:
  - Draw a minibatch of $\boldsymbol{x}$ values from dataset
  - Draw a minibatch of $\boldsymbol{z}$ values from prior (e.g., $\mathcal{N}(\mathbf{0}, I)$)
  - Simultaneously update $\boldsymbol{\theta}^{(G)}$ to reduce $J^{(G)}$ and $\boldsymbol{\theta}^{(D)}$ to reduce $J^{(D)}$, via, e.g., Adam
- For $J^{(D)}$, common to use cross-entropy where label is 1 for real and 0 for fake
- Since generator wants to trick discriminator, can use $J^{(G)} = -J^{(D)}$
  - Others exist that are generally better in practice, e.g., based on ML

Generative Adversarial Network

DCGAN: Radford et al. (2015)

- "Deep, convolution GAN"
- Generator uses **transposed convolutions** (e.g., `tf.layers.conv2d_transpose`) without pooling to upsample images for input to discriminator



Generative Adversarial Network

DCGAN Generated Images: Bedrooms

Trained from LSUN dataset, sampled z space

Generative Adversarial Network

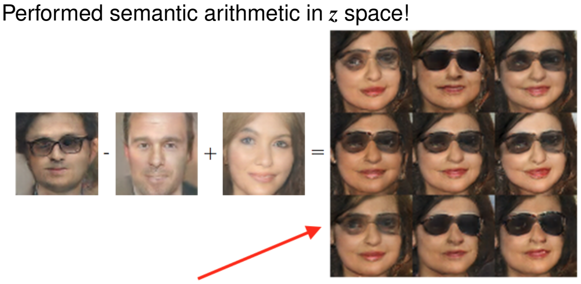
DCGAN Generated Images: Adele Facial Expressions

Trained from frame grabs of interview, sampled z space

Generative Adversarial Network

DCGAN Generated Images: Latent Space Arithmetic

Performed semantic arithmetic in z space!



(Non-center images have noise added in z space; center is noise-free)