Autoencoders
Neural Networks Online Course

- **Disclaimer**: Much of the material and slides for this lecture were borrowed from Hugo Larochelle’s class on Neural Networks:

  http://info.usherbrooke.ca/hlarochelle/neural_networks

- Hugo’s class covers many other topics: convolutional networks, neural language model, Boltzmann machines, autoencoders, sparse coding, etc.

- We will use his material for some of the other lectures.
Autoencoders

- Feed-forward neural network trained to reproduce its input at the output layer

\[
\hat{x} = o(\hat{a}(x)) = \text{sigm}(c + W^*h(x))
\]

Decoder

\[
h(x) = g(a(x)) = \text{sigm}(b + Wx)
\]

Encoder

For binary units
Autoencoders

- Details of what goes inside the encoder and decoder matter!
- Need constraints to avoid learning an identity.
Autoencoders

\[ z = \sigma(Wx) \]

Encoder filters \( W \).

Sigmoid function

\[ \sigma(x) = \frac{1}{1 + \exp(-x)} \]

Input Image \( x \)

Linear function

Decoder filters \( D \)

Binary Features \( z \)
Another Autoencoder Model

Binary Features $z$

$\sigma(W^T z)$

$z = \sigma(Wx)$

Binary Input $x$

Decoder filters $W^T$

Encoder filters $W$.

Sigmoid function

- Need additional constraints to avoid learning an identity.
- Relates to Restricted Boltzmann Machines.
- Encoder and Decoder filters can be different.
Loss Function

- **Loss function** for binary inputs

\[
l(f(x)) = - \sum_k (x_k \log(\hat{x}_k) + (1 - x_k) \log(1 - \hat{x}_k))
\]

  - Cross-entropy error function (reconstruction loss) \( f(x) \equiv \hat{x} \)

- **Loss function** for real-valued inputs

\[
l(f(x)) = \frac{1}{2} \sum_k (\hat{x}_k - x_k)^2
\]

  - sum of squared differences (reconstruction loss)
  - we use a linear activation function at the output
Loss Function

- For both cases, the gradient $\nabla \hat{a}(x^{(t)}) l(f(x^{(t)}))$ has a very simple form:

  $$\nabla \hat{a}(x^{(t)}) l(f(x^{(t)})) = \hat{x}^{(t)} - x^{(t)} \quad f(x) \equiv \hat{x}$$

- **Parameter gradients** are obtained by backpropagating the gradient $\nabla \hat{a}(x^{(t)}) l(f(x^{(t)}))$ like in a regular network

  - important: when using tied weights ($W^* = W^\top$), $\nabla_W l(f(x^{(t)}))$ is the sum of two gradients
  - this is because $W$ is present in the encoder and in the decoder
Autoencoder

• Adapting an autoencoder to a new type of input

  ➢ choose a joint distribution \( p(x|\mu) \) over the inputs, where \( \mu \) is the vector of parameters of that distribution
  ➢ choose the relationship between \( \mu \) and the hidden layer \( h(x) \)
  ➢ use \( l(f(x)) = -\log p(x|\mu) \) as the loss function

• Example: we get the sum of squared distance by

  ➢ choosing a Gaussian distribution with mean \( \mu \) and identity covariance for \( p(x|\mu) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2} \sum_k (x_k - \mu_k)^2\right) \)
  ➢ And choosing \( \mu = c + W^* h(x) \)
Example: MNIST

- MNIST dataset:
Learned Features

- MNIST dataset:

![Image of learned features from MNIST dataset using RBM and Autoencoder]
If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.

• The K hidden units will span the same space as the first k principal components. The weight vectors may not be orthogonal.

• With nonlinear hidden units, we have a nonlinear generalization of PCA.
Optimality of the Linear Autoencoder

• Let us consider the following theorem:

  ➢ let \( \mathbf{A} \) be any matrix, with singular value decomposition \( \mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^\top \)
    • \( \Sigma \) is a diagonal matrix
    • \( \mathbf{V}, \mathbf{U} \) are orthonormal matrices (columns/rows are orthonormal vectors)
Optimality of the Linear Autoencoder

• Let us consider the following theorem:

  ➢ let $\mathbf{A}$ be any matrix, with singular value decomposition $\mathbf{A} = \mathbf{U} \, \Sigma \, \mathbf{V}^\top$
    - $\Sigma$ is a diagonal matrix
    - $\mathbf{V}$, $\mathbf{U}$ are orthonormal matrices (columns/rows are orthonormal vectors)
  ➢ let $\mathbf{U}_{:, \leq k} \, \Sigma_{\leq k, \leq k} \, \mathbf{V}_{:, \leq k}^\top$ be the decomposition where we keep only the $k$ largest singular values
  ➢ then, the matrix $\mathbf{B}$ of rank $k$ that is closest to $\mathbf{A}$:
    $$\mathbf{B}^* = \arg \min_{\mathbf{B} \text{ s.t. } \text{rank}(\mathbf{B})=k} \| \mathbf{A} - \mathbf{B} \|_F$$
    is $\mathbf{B}^* = \mathbf{U}_{:, \leq k} \, \Sigma_{\leq k, \leq k} \, \mathbf{V}_{:, \leq k}^\top$
\[
\min \sum_{t} \frac{1}{2} \sum_{i} (x_i^{(t)} - \hat{x}_i^{(t)})^2 \geq \min \frac{1}{2} \| \mathbf{X} - \mathbf{W}^* \mathbf{h(X)} \|_F^2
\]

matrix where columns are \( \mathbf{X}^{(t)} \)
based on linear encoder

\[
\arg \min_{\mathbf{W}^*, \mathbf{h(X)}} \frac{1}{2} \| \mathbf{X} - \mathbf{W}^* \mathbf{h(X)} \|_F^2 \quad = \quad (\mathbf{W}^* \leftarrow \mathbf{U}_{:, \leq k} \Sigma_{\leq k, \leq k}, \mathbf{h(X)} \leftarrow \mathbf{V}_{:, \leq k})
\]

based on previous theorem, where \( \mathbf{X} = \mathbf{U} \Sigma \mathbf{V}^T \)

and \( k \) is the hidden layer size

Let's show \( \mathbf{h(X)} \) is a linear encoder:

\[
\mathbf{h(X)} = \mathbf{V}_{:, \leq k}^T
\]

\[
= \mathbf{V}_{:, \leq k}^T (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X})
\]

\[
= \mathbf{V}_{:, \leq k}^T (\mathbf{V} \Sigma^T \mathbf{U}^T \mathbf{U} \Sigma \mathbf{V}^T)^{-1} (\mathbf{V} \Sigma^T \mathbf{U}^T \mathbf{X})
\]

\[
= \mathbf{V}_{:, \leq k}^T \mathbf{V} (\Sigma^T \Sigma)^{-1} \mathbf{V}^T \mathbf{V} \Sigma^T \mathbf{U}^T \mathbf{X}
\]

\[
= \mathbf{V}_{:, \leq k}^T \mathbf{V} (\Sigma^T \Sigma)^{-1} \Sigma^T \mathbf{U}^T \mathbf{X}
\]

\[
= \mathbf{I}_{\leq k, \cdot} \cdot (\Sigma^T \Sigma)^{-1} \Sigma^T \mathbf{U}^T \mathbf{X}
\]

\[
= \mathbf{I}_{\leq k, \cdot} \cdot \Sigma^{-1} (\Sigma^T)^{-1} \Sigma^T \mathbf{U}^T \mathbf{X}
\]

\[
= \mathbf{I}_{\leq k, \cdot} \cdot \Sigma^{-1} \mathbf{U}^T \mathbf{X}
\]

\[
= \Sigma_{\leq k, \leq k} (\mathbf{U}_{:, \leq k})^T \mathbf{X}
\]

this is a linear encoder

(Slide from Hugo Larochelle)
Optimality of the Linear Autoencoder

So an optimal pair of encoder and decoder is

\[
\begin{align*}
\hat{x} &= (U_{\leq k} \sum_{\leq k} h(x)) x \\
\hat{x} &= (U_{\leq k} \sum_{\leq k} h(x)) x \\
\end{align*}
\]

- for the sum of squared difference error
- for an autoencoder with a linear decoder
- where optimality means “has the lowest training reconstruction error”
Optimality of the Linear Autoencoder

• So an optimal pair of encoder and decoder is

\[ h(x) = \left( \sum_{k \leq k}^{-1} (U_{\cdot, \leq k})^\top \right) x \]

\[ \hat{x} = (U_{\cdot, \leq k} \sum_{k \leq k} \cdot) h(x) \]

• If inputs are normalized as follows:

\[ x^{(t)} \leftarrow \frac{1}{\sqrt{T}} \left( x^{(t)} - \frac{1}{T} \sum_{t'=1}^{T} x^{(t')} \right) \]

- encoder corresponds to Principal Component Analysis (PCA)
- singular values and (left) vectors = the eigenvalues/vectors of covariance matrix
Undercomplete Representation

- Hidden layer is undercomplete if smaller than the input layer (bottleneck layer, e.g. dimensionality reduction):
  - hidden layer “compresses” the input
  - will compress well only for the training distribution

- Hidden units will be
  - good features for the training distribution
  - will not be robust to other types of input
Overcomplete Representation

• Hidden layer is **overcomplete** if greater than the input layer

  - no compression in hidden layer
  - each hidden unit could copy a different input component

• No guarantee that the hidden units will extract **meaningful structure**
Denoising Autoencoder

• **Idea**: representation should be robust to introduction of noise:
  - random assignment of subset of inputs to 0, with probability \( \nu \)
  - Similar to dropouts on the input layer
  - Gaussian additive noise

• **Reconstruction** \( \hat{\mathbf{x}} \) computed from the corrupted input \( \tilde{\mathbf{x}} \)

• **Loss function** compares \( \hat{\mathbf{x}} \) reconstruction with the noiseless input \( \mathbf{x} \)

(Vincent et. al., ICML 2008)
\[ \hat{x} = \text{sigm}(c + W^* h(\tilde{x})) \]
Learned Filters

Non-corrupted

25% corrupted input

(Vincent et.al., ICML 2008)
Learned Filters

Non-corrupted

50% corrupted input

(Vincent et.al., ICML 2008)
Squared Error Loss

- Training on natural image patches, with squared loss
  - PCA may not the best solution
Squared Error Loss

- Training on natural image patches, with squared loss
  - Not equivalent to weight decay
Contractive Autoencoders

- Alternative approach to avoid uninteresting solutions
  - add an explicit term in the loss that penalizes that solution

- We wish to extract features that only reflect variations observed in the training set
  - we'd like to be invariant to the other variations

(Salah Rifai et al., 2011)
Contractive Autoencoders

• Consider the following loss function:

\[
l(f(x^{(t)})) + \lambda \left\| \nabla_{x^{(t)}} h(x^{(t)}) \right\|_F^2
\]

Reconstruction Loss

\[
l(f(x^{(t)})) = - \sum_k \left( x_k^{(t)} \log(\hat{x}_k^{(t)}) + (1 - x_k^{(t)}) \log(1 - \hat{x}_k^{(t)}) \right)
\]

For the binary observations:

\[
\left\| \nabla_{x^{(t)}} h(x^{(t)}) \right\|_F^2 = \sum_j \sum_k \left( \frac{\partial h(x^{(t)}_j)}{\partial x_k^{(t)}} \right)^2
\]

Autoencoder attempts to preserve all information

Encoder throws away all information
Contractive Autoencoders

• Illustration:

encoder doesn't need to be sensitive to this variation (not observed in training set)

encoder must be sensitive to this variation to reconstruct well
Pros and Cons

• Advantage of denoising autoencoder: simpler to implement
  - requires adding one or two lines of code to regular autoencoder
  - no need to compute Jacobian of hidden layer

• Advantage of contractive autoencoder: gradient is deterministic
  - can use second order optimizers (conjugate gradient, LBFGS, etc.)
  - might be more stable than denoising autoencoder, which uses a sampled gradient
Autoencoder

- Details of what goes inside the encoder and decoder matter!
- Need constraints to avoid learning an identity.
Predictive Sparse Decomposition

At training time:

\[
\min_{D,W,z} \left\| Dz - x \right\|^2_2 + \lambda |z|_1 + \left\| \sigma(Wx) - z \right\|^2_2
\]
Stacked Autoencoders

Class Labels

Decoder

Encoder

Features

Sparsity

Decoder

Encoder

Features

Sparsity

Decoder

Encoder

Input x

32
Stacked Autoencoders

Class Labels

Decoder

Encoder

Features

Sparsity

Decoder

Encoder

Features

Sparsity

Input x

Greedy Layer-wise Learning.
Stacked Autoencoders

- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Parameters can be fine-tuned using backpropagation.
Stacked Autoencoders

- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Parameters can be fine-tuned using backpropagation.

Top-down vs. bottom up? Is there a more rigorous mathematical formulation?
Deep Autoencoders
Deep Autoencoders

• We used 25x25 – 2000 – 1000 – 500 – 30 autoencoder to extract 30-D real-valued codes for Olivetti face patches.

• **Top**: Random samples from the test dataset.
• **Middle**: Reconstructions by the 30-dimensional deep autoencoder.
• **Bottom**: Reconstructions by the 30-dimentinoal PCA.
• The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into 402,207 training and 402,207 test).

• “Bag-of-words” representation: each article is represented as a vector containing the counts of the most frequently used 2000 words in the training set.

(Hinton and Salakhutdinov, Science 2006)
Information Retrieval

Reuters Dataset

Reuters dataset: 804,414 newswire stories.

Deep generative model significantly outperforms LSA and LDA topic models
Semantic Hashing

- Learn to map documents into semantic 20-D binary codes.
- Retrieve similar documents stored at the nearby addresses with no search at all.

(Salakhutdinov and Hinton, SIGIR 2007)
Searching Large Image Database using Binary Codes

- Map images into binary codes for fast retrieval.

- Small Codes, Torralba, Fergus, Weiss, CVPR 2008
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 2011
- Norouzi and Fleet, ICML 2011
Learning Similarity Measures

- Learn a nonlinear transformation of the input space.
- Optimize to make KNN perform well in the low-dimensional feature space

(Salakhutdinov and Hinton, AI and Statistics 2007)
Learning Similarity Measures

Neighborhood Component Analysis

Linear discriminant Analysis

PCA

Learning Similarity Metric

\[ D[y^a, y^b] \]
Learning Similarity Measures

Learning Similarity Metric

\[ D[y^a, y^b] \]

- As we change unit 25 in the code layer, ``3'' image turns into ``5'' image.

- As we change unit 42 in the code layer, thick ``3'' image turns into skinny ``3''.

\[ y^a \quad 30 \quad 2000 \quad 500 \quad 500 \quad w_1 \]

\[ y^b \quad 30 \quad 2000 \quad 500 \quad 500 \quad w_1 \]

\[ X^a \quad X^b \]
Learning Invariant Features of Tumor Signature

A viable tumor region

Mixed necrotic and apoptotic regions

(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.)
Reconstruction Independent Subspace Analysis (RISA)

\[ p_i(x; W, V) = \sqrt{\sum_{k=1}^{m} V_{ik} \left( \sum_{j=1}^{d} W_{kj} x_i \right)^2} \]

Total input into the 1st layer.

(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.)
Reconstruction Independent Subspace Analysis (RISA)

• Given a set of training patches: \( \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\} \), we minimize:

\[
\min_W \sum_{n=1}^{N} \left( \sum_{i=1}^{m} p_i(x^{(n)}; W, V) + \lambda \|W W^T x^{(n)} - x^{(n)} \|^2 \right)
\]

Reconstruction term

(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.)
RISA features work much better for classification compared to hand-crafted features.

(Le, Han, Spellman, Borowsky, Parvin, ISBI 2012.)