Autoencoders

Jian Li
2017-10-17
Contents

• Stochastic Autoencoders
• Denoising Autoencoders
• Contractive Autoencoders
• Application of Autoencoders
Stochastic autoencoders

Usual autoencoder:
- $x \rightarrow h$
- $h \rightarrow x$

Stochastic encoder:
- $x \rightarrow p_{encoder}(h|x)$

Stochastic decoder:
- $h \rightarrow p_{decoder}(x|h)$

Stochastic autoencoder may be estimated with maximum likelihood.
Stochastic autoencoders

• $p_{\text{decoder}}(x|h)$ determines the **output units** and the **loss function**.

• Train the autoencoder by minimizing $-\log p_{\text{decoder}}(x|h)$

• E.g., $p_{\text{decoder}}(x|h) = N(x; \hat{x}(h), \sigma^2)$

• Since the examples are assumed to be i.i.d., the loss is

$$-\log p_{\text{decoder}}(x|h) = -\sum_{i=1}^{m} \log p_{\text{decoder}}(x^i|h^i)$$

$$= m \log \sigma + \frac{m}{2} \log(2\pi) + \sum_{i=1}^{m} \frac{\|\hat{x}^i - x^i\|^2}{2\sigma^2}$$

• The MSE error: $\text{MSE}_{\text{decoder}} = \frac{1}{m} \sum_{i=1}^{m} \|\hat{x}^i - x^i\|^2$
Denoising autoencoders

- Solved task:
  \[ \sum_{n=1}^{N} \mathcal{L}(x, g_{\theta}(f_{\theta}(\tilde{x}))) \rightarrow \min_{\theta} \]
  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 autoencoders

The autoencoder learns a reconstruction distribution \( p_{\text{reconstruct}}(x|\tilde{x}) \) estimated from training pairs \((x, \tilde{x})\), as follows:

1. Sample a training example \( x \) from the training data.
2. Sample a corrupted version \( \tilde{x} \) from \( C(\tilde{x} | x = x) \).
3. Use \((x, \tilde{x})\) as a training example for estimating the autoencoder reconstruction distribution \( p_{\text{reconstruct}}(x | \tilde{x}) = p_{\text{decoder}}(x | h) \) with \( h \) the output of encoder \( f(\tilde{x}) \) and \( p_{\text{decoder}} \) typically defined by a decoder \( g(h) \).

If encoder is deterministic, \( L = -\log p_{\text{decoder}}(x | h = f(\tilde{x})) \)
What it learns

- Gray circle: corruption area
- Black line: manifold, where objects are concentrated.
- Red crosses: training set.
- Green lines: $g(f(x)) - x$
Contractive autoencoders

- minimize $L(x, g(f(x))) + \Omega(h, x)$,

$$\Omega(h) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2. \tag{14.18}$$

The penalty $\Omega(h)$ is the squared Frobenius norm (sum of squared elements) of the Jacobian matrix of partial derivatives associated with the encoder function.

- Jacobian reduces sensitivity of $h$ to $x$
- Such regularization↑robustness of representation to small variations in $x$.
- $\lambda$ controls tradeoff between reconstruction error and robustness.
Manifold learning

- Reconstruction error alone: learn an identity function.
- Contractive penalty alone: learn features that are constant w.r.t. input $x$.
- The compromise between the two forces makes representation $h$ only sensitive to changes along the manifold directions.

PCA example

Non-linear Manifold
Contractive autoencoders

• Manifold learning

  - $h = f(x)$ maps input to coordinates in embedded space.
  - $f(x + \Delta x) \approx f(x) + J_f(x)\Delta x$
  - Directions $\Delta x$:
    - 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

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 \( f(x) \).

\(^1\text{See Alain and Bengio (2013).}\)
Application of autoencoders

Applications:
- dimensionality reduction
  - visualization
  - feature extraction
  - ↑ prediction accuracy
  - ↑ speed of prediction
  - ↓ memory requirements
- semantic hashing
- unsupervised pretraining
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
Semantic hashing

- 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.