7.2. Autoencoders

François Fleuret
https://fleuret.org/dlc/
Nov 24, 2020
Many applications such as image synthesis, denoising, super-resolution, speech synthesis, compression, etc. require to go beyond classification and regression, and model explicitly a high dimension signal.
Many applications such as image synthesis, denoising, super-resolution, speech synthesis, compression, etc. require to go beyond classification and regression, and model explicitly a high dimension signal.

This modeling consists of finding “meaningful degrees of freedom” that describe the signal, and are of lesser dimension.
Original space $\mathcal{X}$
Original space $\mathcal{X}$
Original space $\mathcal{X}$

Latent space $\mathcal{F}$
Original space $\mathcal{X}$

Latent space $\mathcal{F}$

function $f$
Original space $\mathcal{X}$

Latent space $\mathcal{F}$
Original space $\mathcal{X}$

Latent space $\mathcal{F}$
Original space $\mathcal{X}$
When dealing with real-world signals, this objective involves the same theoretical and practical issues as for classification or regression: defining the right class of high-dimension models, and optimizing them.

Regarding synthesis, we saw that deep feed-forward architectures exhibit good generative properties, which motivates their use explicitly for that purpose.
Autoencoders
An autoencoder maps a space to itself and is [close to] the identity on the data.

Dimension reduction can be achieved with an autoencoder composed of an **encoder** $f$ from the original space $\mathcal{X}$ to a **latent** space $\mathcal{F}$, and a **decoder** $g$ to map back to $\mathcal{X}$ (Bourlard and Kamp, 1988; Hinton and Zemel, 1994).
An autoencoder maps a space to itself and is [close to] the identity on the data.

Dimension reduction can be achieved with an autoencoder composed of an encoder $f$ from the original space $\mathcal{X}$ to a latent space $\mathcal{F}$, and a decoder $g$ to map back to $\mathcal{X}$ (Bourlard and Kamp, 1988; Hinton and Zemel, 1994).
An autoencoder maps a space to itself and is [close to] the identity on the data.

Dimension reduction can be achieved with an autoencoder composed of an encoder $f$ from the original space $\mathcal{X}$ to a latent space $\mathcal{F}$, and a decoder $g$ to map back to $\mathcal{X}$ (Bourlard and Kamp, 1988; Hinton and Zemel, 1994).
An autoencoder maps a space to itself and is [close to] the identity on the data.

Dimension reduction can be achieved with an autoencoder composed of an **encoder** \( f \) from the original space \( \mathcal{X} \) to a **latent** space \( \mathcal{F} \), and a **decoder** \( g \) to map back to \( \mathcal{X} \) (Bourlard and Kamp, 1988; Hinton and Zemel, 1994).
An autoencoder maps a space to itself and is [close to] the identity on the data.

Dimension reduction can be achieved with an autoencoder composed of an encoder $f$ from the original space $\mathcal{X}$ to a latent space $\mathcal{F}$, and a decoder $g$ to map back to $\mathcal{X}$ (Bourlard and Kamp, 1988; Hinton and Zemel, 1994).

If the latent space is of lower dimension, the autoencoder has to capture a "good" parametrization, and in particular dependencies between components.
Let $q$ be the data distribution over $\mathcal{X}$. A good autoencoder could be characterized with the quadratic loss

$$\mathbb{E}_{X \sim q}[\|X - g \circ f(X)\|^2] \simeq 0.$$
Let $q$ be the data distribution over $\mathcal{X}$. A good autoencoder could be characterized with the quadratic loss

$$
\mathbb{E}_{X \sim q} \left[ \| X - g \circ f(X) \|^2 \right] \approx 0.
$$

Given two parametrized mappings $f(\cdot; w_f)$ and $g(\cdot; w_g)$, training consists of minimizing an empirical estimate of that loss

$$
\hat{w}_f, \hat{w}_g = \arg\min_{w_f, w_g} \frac{1}{N} \sum_{n=1}^{N} \| x_n - g(f(x_n; w_f); w_g) \|^2.
$$
Let $q$ be the data distribution over $\mathcal{X}$. A good autoencoder could be characterized with the quadratic loss

$$
\mathbb{E}_{X \sim q} \left[ \| X - g \circ f(X) \|^2 \right] \approx 0.
$$

Given two parametrized mappings $f(\cdot; w_f)$ and $g(\cdot; w_g)$, training consists of minimizing an empirical estimate of that loss

$$
\hat{w}_f, \hat{w}_g = \arg\min_{w_f, w_g} \frac{1}{N} \sum_{n=1}^{N} \| x_n - g(f(x_n; w_f); w_g) \|^2.
$$

A simple example of such an autoencoder would be with both $f$ and $g$ linear, in which case the optimal solution is given by PCA.
Let $q$ be the data distribution over $\mathcal{X}$. A good autoencoder could be characterized with the quadratic loss

$$\mathbb{E}_{X \sim q} \left[ \| X - g \circ f(X) \|^2 \right] \approx 0.$$ 

Given two parametrized mappings $f(\cdot; w_f)$ and $g(\cdot; w_g)$, training consists of minimizing an empirical estimate of that loss

$$\hat{w}_f, \hat{w}_g = \arg\min_{w_f, w_g} \frac{1}{N} \sum_{n=1}^{N} \| x_n - g(f(x_n; w_f); w_g) \|^2.$$ 

A simple example of such an autoencoder would be with both $f$ and $g$ linear, in which case the optimal solution is given by PCA. Better results can be achieved with more sophisticated classes of mappings, in particular deep architectures.
Deep Autoencoders
A deep autoencoder combines an encoder composed of convolutional layers, with a decoder composed of transposed convolution or other interpolating layers. *E.g.* for MNIST:

```python
AutoEncoder (
  (encoder): Sequential (
    (0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU (inplace)
    (2): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1))
    (3): ReLU (inplace)
    (4): Conv2d(32, 32, kernel_size=(4, 4), stride=(2, 2))
    (5): ReLU (inplace)
    (6): Conv2d(32, 32, kernel_size=(4, 4), stride=(2, 2))
    (7): ReLU (inplace)
    (8): Conv2d(32, 8, kernel_size=(4, 4), stride=(1, 1))
  )

  (decoder): Sequential (
    (0): ConvTranspose2d(8, 32, kernel_size=(4, 4), stride=(1, 1))
    (1): ReLU (inplace)
    (2): ConvTranspose2d(32, 32, kernel_size=(3, 3), stride=(2, 2))
    (3): ReLU (inplace)
    (4): ConvTranspose2d(32, 32, kernel_size=(4, 4), stride=(2, 2))
    (5): ReLU (inplace)
    (6): ConvTranspose2d(32, 32, kernel_size=(5, 5), stride=(1, 1))
    (7): ReLU (inplace)
    (8): ConvTranspose2d(32, 1, kernel_size=(5, 5), stride=(1, 1))
  )
)
Encoder

Tensor sizes / operations

1 × 28 × 28

\texttt{nn.Conv2d(1, 32, kernel\_size=5, stride=1)}

32 × 24 × 24

\texttt{nn.Conv2d(32, 32, kernel\_size=5, stride=1)}

32 × 20 × 20

\texttt{nn.Conv2d(32, 32, kernel\_size=4, stride=2)}

32 × 9 × 9

\texttt{nn.Conv2d(32, 32, kernel\_size=3, stride=2)}

32 × 4 × 4

\texttt{nn.Conv2d(32, 8, kernel\_size=4, stride=1)}

8 × 1 × 1
## Decoder

### Tensor sizes / operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>$8 \times 1 \times 1$</td>
<td>$8 \times 1 \times 1$</td>
</tr>
<tr>
<td><code>nn.ConvTranspose2d(8, 32, kernel_size=4, stride=1)</code></td>
<td>$32 \times 4 \times 4$</td>
</tr>
<tr>
<td><code>nn.ConvTranspose2d(32, 32, kernel_size=3, stride=2)</code></td>
<td>$32 \times 9 \times 9$</td>
</tr>
<tr>
<td><code>nn.ConvTranspose2d(32, 32, kernel_size=4, stride=2)</code></td>
<td>$32 \times 20 \times 20$</td>
</tr>
<tr>
<td><code>nn.ConvTranspose2d(32, 32, kernel_size=5, stride=1)</code></td>
<td>$32 \times 24 \times 24$</td>
</tr>
<tr>
<td><code>nn.ConvTranspose2d(32, 1, kernel_size=5, stride=1)</code></td>
<td>$1 \times 28 \times 28$</td>
</tr>
</tbody>
</table>
Training is achieved with quadratic loss and Adam

```python
model = AutoEncoder(nb_channels, embedding_dim)

optimizer = optim.Adam(model.parameters(), lr = 1e-3)

for epoch in range(args.nb_epochs):
    for input in train_input.split(batch_size):
        z = model.encode(input)
        output = model.decode(z)
        loss = 0.5 * (output - input).pow(2).sum() / input.size(0)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```
$X$ (original samples)

$g \circ f(X)$ (CNN, $d = 2$)

$g \circ f(X)$ (PCA, $d = 2$)
\[ X \text{ (original samples)} \]

\[ g \circ f(X) \text{ (CNN, } d = 4) \]

\[ g \circ f(X) \text{ (PCA, } d = 4) \]
$g \circ f(X)$ (CNN, $d = 8$)

$g \circ f(X)$ (PCA, $d = 8$)
$g \circ f(X)$ (CNN, $d = 16$)

$g \circ f(X)$ (PCA, $d = 16$)
$X$ (original samples)

$g \circ f(X)$ (CNN, $d = 32$)

$g \circ f(X)$ (PCA, $d = 32$)
To get an intuition of the latent representation, we can pick two samples \( x \) and \( x' \) at random and interpolate samples along the line in the latent space

\[
\forall x, x' \in \mathcal{X}^2, \ \alpha \in [0, 1], \ \xi(x, x', \alpha) = g((1 - \alpha)f(x) + \alpha f(x')).
\]
To get an intuition of the latent representation, we can pick two samples $x$ and $x'$ at random and interpolate samples along the line in the latent space

$$\forall x, x' \in \mathcal{X}^2, \alpha \in [0, 1], \quad \xi(x, x', \alpha) = g((1 - \alpha)f(x) + \alpha f(x')).$$
To get an intuition of the latent representation, we can pick two samples $x$ and $x'$ at random and interpolate samples along the line in the latent space

$$\forall x, x' \in \mathcal{X}^2, \alpha \in [0, 1], \quad \xi(x, x', \alpha) = g((1 - \alpha)f(x) + \alpha f(x')).$$
To get an intuition of the latent representation, we can pick two samples $x$ and $x'$ at random and interpolate samples along the line in the latent space

$$\forall x, x' \in \mathcal{X}^2, \alpha \in [0, 1], \quad \xi(x, x', \alpha) = g((1 - \alpha)f(x) + \alpha f(x'))$$.
PCA interpolation ($d = 32$)
Autoencoder interpolation \((d = 8)\)
Autoencoder interpolation \((d = 32)\)
And we can assess the generative capabilities of the decoder $g$ by introducing a [simple] density model $q^Z$ over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with $g$. 
And we can assess the generative capabilities of the decoder \( g \) by introducing a [simple] density model \( q^Z \) over the latent space \( \mathcal{F} \), sample there, and map the samples into the image space \( \mathcal{X} \) with \( g \).

We can for instance use a Gaussian model with diagonal covariance matrix.

\[
f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})
\]

where \( \hat{m} \) is a vector and \( \hat{\Delta} \) a diagonal matrix, both estimated on training data.
And we can assess the generative capabilities of the decoder $g$ by introducing a [simple] density model $q^Z$ over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with $g$.

We can for instance use a Gaussian model with diagonal covariance matrix.

$$f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})$$

where $\hat{m}$ is a vector and $\hat{\Delta}$ a diagonal matrix, both estimated on training data.
And we can assess the generative capabilities of the decoder $g$ by introducing a [simple] density model $q_{Z}$ over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with $g$.

We can for instance use a Gaussian model with diagonal covariance matrix.

$$f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})$$

where $\hat{m}$ is a vector and $\hat{\Delta}$ a diagonal matrix, both estimated on training data.
And we can assess the generative capabilities of the decoder $g$ by introducing a [simple] density model $q^Z$ over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with $g$.

We can for instance use a Gaussian model with diagonal covariance matrix.

$$f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})$$

where $\hat{m}$ is a vector and $\hat{\Delta}$ a diagonal matrix, both estimated on training data.
And we can assess the generative capabilities of the decoder $g$ by introducing a [simple] density model $q^Z$ over the latent space $\mathcal{F}$, sample there, and map the samples into the image space $\mathcal{X}$ with $g$.

We can for instance use a Gaussian model with diagonal covariance matrix.

$$f(X) \sim \mathcal{N}(\hat{m}, \hat{\Delta})$$

where $\hat{m}$ is a vector and $\hat{\Delta}$ a diagonal matrix, both estimated on training data.
Autoencoder sampling ($d = 8$)

Autoencoder sampling ($d = 16$)

Autoencoder sampling ($d = 32$)
These results are unsatisfying, because the density model used on the latent space $\mathcal{F}$ is too simple and inadequate.

Building a “good” model amounts to our original problem of modeling an empirical distribution, although it may now be in a lower dimension space.
The end
References
