Recall: Encoder Networks

- Identity mapping through a bottleneck
- Also called N–M–N task
- Used to investigate hidden unit representations

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>10000</td>
</tr>
<tr>
<td>01000</td>
<td>01000</td>
</tr>
<tr>
<td>00100</td>
<td>00100</td>
</tr>
<tr>
<td>00010</td>
<td>00010</td>
</tr>
<tr>
<td>00001</td>
<td>00001</td>
</tr>
</tbody>
</table>

Autoencoder Networks

- Output is trained to reproduce the input as closely as possible
- Activations normally pass through a bottleneck, so the network is forced to compress the data in some way
- Like the RBM, autoencoders can be used to automatically extract abstract features from the input

Outline

- Autoencoder Networks (14.1)
- Regularized Autoencoders (14.2)
- Stochastic Encoders and Decoders (14.4)
- Generative Models
- Variational Autoencoders (20.10.3)
Autoencoder Networks

<table>
<thead>
<tr>
<th>input</th>
<th>hidden</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the encoder computes \( z = f(x) \) and the decoder computes \( g(f(x)) \) then we aim to minimize some distance function between \( x \) and \( g(f(x)) \)

\[
E = L(x, g(f(x)))
\]

Autoencoder as Pretraining

- after an autoencoder is trained, the decoder part can be removed and replaced with, for example, a classification layer
- this new network can then be trained by backpropagation
- the features learned by the autoencoder then serve as initial weights for the supervised learning task

Greedy Layerwise Pretraining

- Autoencoders can be used as an alternative to Restricted Boltzmann Machines, for greedy layerwise pretraining.
- An autoencoder with one hidden layer is trained to reconstruct the inputs. The first layer (encoder) of this network becomes the first layer of the deep network.
- Each subsequent layer is then trained to reconstruct the previous layer.
- A final classification layer is then added to the resulting deep network, and the whole thing is trained by backpropagation.

Avoiding Trivial Identity

- If there are more hidden nodes than inputs (which often happens in image processing) there is a risk the network may learn a trivial identity mapping from input to output.
- We generally try to avoid this by introducing some form of regularization.
### Regularized Autoencoders (14.2)

- autoencoders with dropout at hidden layer(s)
- sparse autoencoders
- contractive autoencoders
- denoising autoencoders

### Sparse Autoencoder (14.2.1)

- One way to regularize an autoencoder is to include a penalty term in the loss function, based on the hidden unit activations.
- This is analogous to the weight decay term we previously used for supervised learning.
- One popular choice is to penalize the sum of the absolute values of the activations in the hidden layer

\[
E = L(x, g(f(x))) + \lambda \sum_i |h_i|
\]

- This is sometimes known as $L_1$-regularization (because it involves the absolute value rather than the square); it can encourage some of the hidden units to go to zero, thus producing a sparse representation.

### Contractive Autoencoder (14.2.3)

- Another popular penalty term is the $L_2$-norm of the derivatives of the hidden units with respect to the inputs

\[
E = L(x, g(f(x))) + \lambda \sum_i \|\nabla x h_i\|^2
\]

- This forces the model to learn hidden features that do not change much when the training inputs $x$ are slightly altered.

### Denoising Autoencoder (14.2.2)

Another regularization method, similar to contractive autoencoder, is to add noise to the inputs, but train the network to recover the original input

repeat:
  - sample a training item $x^{(i)}$
  - generate a corrupted version $\tilde{x}$ of $x^{(i)}$
  - train to reduce $E = L(x^{(i)}, g(f(\tilde{x})))$
end
Generative Models

- Sometimes, as well as reproducing the training items \( \{ x^{(i)} \} \), we also want to be able to use the decoder to generate new items which are of a similar “style” to the training items.
- In other words, we want to be able to choose latent variables \( z \) from a standard Normal distribution \( p(\mu) \), feed these values of \( z \) to the decoder, and have it produce a new item \( x \) which is somehow similar to the training items.
- Generative models can be:
  - explicit (Variational Autoencoders)
  - implicit (Generative Adversarial Networks)

Gaussian Distribution (3.9.3)

\[
P_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

- \( \mu = \) mean
- \( \sigma = \) standard deviation
- Multivariate Gaussian: \( P_{\mu,\sigma}(x) = \prod_i P_{\mu_i,\sigma_i}(x_i) \)
Entropy and KL-Divergence

- The entropy of a distribution $q()$ is $H(q) = \int q(\theta)(-\log q(\theta))d\theta$
- In Information Theory, $H(q)$ is the amount of information (bits) required to transmit a random sample from distribution $q()$
- For a Gaussian distribution, $H(q) = \sum \log \sigma_i$
- KL-Divergence $D_{KL}(q \parallel p) = \int q(\theta)(\log q(\theta) - \log p(\theta))d\theta$
- $D_{KL}(q \parallel p)$ is the number of extra bits we need to transmit if we designed a code for $p()$ but the samples are drawn from $q()$ instead.
- If $p(z)$ is Standard Normal distribution, minimizing $D_{KL}(q_\phi(z) || p(z))$ encourages $q_\phi()$ to center on zero and spread out to approximate $p()$.

Variational Autoencoder (20.10.3)

Instead of producing a single $z$ for each $x^{(i)}$, the encoder (with parameters $\phi$) can be made to produce a mean $\mu_{\phi}(z^{(i)})$ and standard deviation $\sigma_{\phi}(z^{(i)})$
This defines a conditional (Gaussian) probability distribution $q_\phi(z|x^{(i)})$
We then train the system to maximize
$$\mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}|z) \right] - D_{KL}(q_\phi(z|x^{(i)}) || p(z))$$
- the first term enforces that any sample $z$ drawn from the conditional distribution $q_\phi(z|x^{(i)})$ should, when fed to the decoder, produce something approximating $x^{(i)}$
- the second term encourages $q_\phi(z|x^{(i)})$ to approximate $p(z)$
- in practice, the distributions $q_\phi(z|x^{(i)})$ for various $x^{(i)}$ will occupy complementary regions within the overall distribution $p(z)$
Variational Autoencoder Faces

![Variational Autoencoder Faces](image)

Variational Autoencoder

- Variational Autoencoder produces reasonable results
- Tends to produce blurry images
- Often end up using only a small number of the dimensions available to $z$

References

http://kvfrans.com/variational-autoencoders-explained/