Representation Learning

Learning representations of the data that make it easier to extract useful information when building classifiers or other predictors.

Y. Bengio, A. Courville, P. Vincent,
Representation Learning: A Review and New Perspectives,
IEEE Transactions on Software Engineering, 2013.

Data Transformation (Traditional)

Transform your data to facilitate model fitting, discrimination, etc.

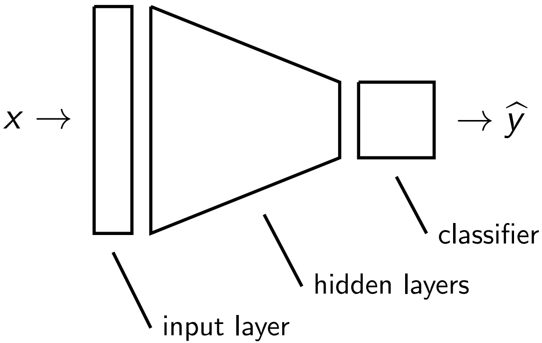
- log-tranformation: $(x, y) \mapsto (\log x, y)$
- log-log-transformation: $(x, y) \mapsto (\log x, \log y)$
- Polynomial features: $(x, y) \mapsto \left((x, x^2, \ldots, x^n), y\right)$
- Basis functions: $(x, y) \mapsto \big(\big(b_1(x), b_2(x), \ldots, b_n(x)\big), y\big)$

Transformations choice informed by *domain knowledge*, *exploratory data analysis*, and *model selection criteria*.

Not *learned from data*.

Supervised Representation Learning

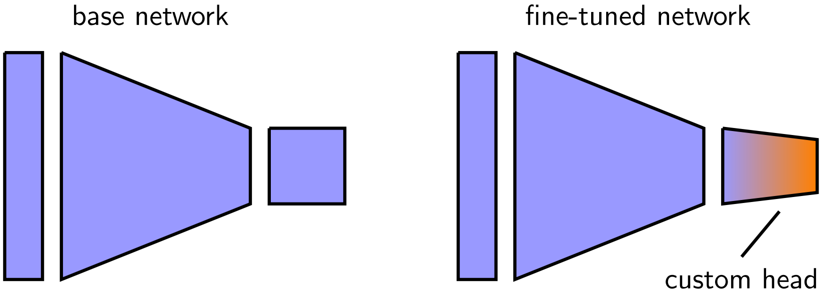
Hidden layers in *deep, supervised models* do representation learning, feed learned representation to, e.g., a classifier.



Transfer Learning

Typically, representations learned by hidden layers are sufficiently general to facilitate *transfer learning*, i.e., application to a different downstream task on a different — often much smaller — dataset.

Fine-tuning: Training a *custom head* on the representation computed by a pretrained *base network*.



Self-Supervised Representation Learning

Learn useful representations of *unlabelled data* using a *pretext task* for which labels can be easily constructed.

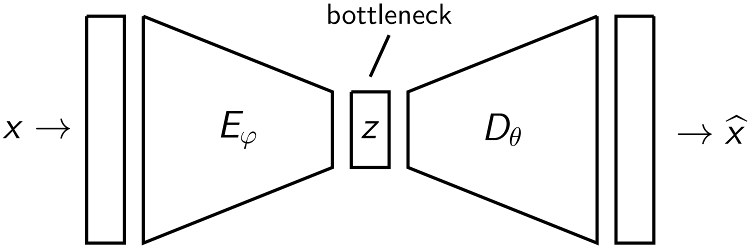
- Compression/Decompression (e.g., *autoencoders*)
- Denoising (e.g., *denoising autoencoders*)
- Image colorization
- Image inpainting
- Jigsaw puzzle solving

Use representation computed in solving pretext task for *downstream tasks*.

Autoencoders

- *data space*, $\mathcal{X}$: $\dim \mathcal{X} \gg 0$
- *code space*, $\mathcal{Z}$: $\dim \mathcal{Z} \ll \dim \mathcal{X}$
- *encoding map*, $E_\varphi : \mathcal{X} \to \mathcal{Z}$
- *decoding map*, $D_\theta : \mathcal{Z} \to \mathcal{X}$
- *reconstruction:* $x \approx D_\theta(E_\varphi(x)) =: \widehat{x}$

Autoencoders



- E_φ and D_θ are typically neural networks.
- Weights φ, θ are learned to minimize *reconstruction error*:

$$(\varphi^*, \theta^*) = \underset{(\varphi,\theta)}{\operatorname{argmin}} \, \mathbb{E}\left[\|\widehat{x} - x\|^2\right]$$

- *Bottleneck* forces the representation z to retain only essential information about x.

Application: Outlier Detection

If x is an *outlier*, i.e., doesn't resemble an element of $\mathcal{X}$, $\widehat{x}$ will likely be a poor reconstruction of x.

Threshold reconstruction error to flag possible outliers.

This is useful for detecting rare events, e.g., in fraud detection.

Standard supervised learning techniques are difficult to apply to datasets with high *class imbalance*.

What about *rare diseases*?

Autoencoders: Limitations

The representations learned by autoencoders aren't particularly useful for downstream tasks.

- To learn a useful representation, a pretext task has to be *semantically meaningful*.
- Pixel-by-pixel reconstruction loss doesn't encourage learning high level features of images.
- As z' moves away from $z = E_\varphi(x)$, $D_\theta(z')$ diverges visually from $D_\theta(z)$ very quickly.

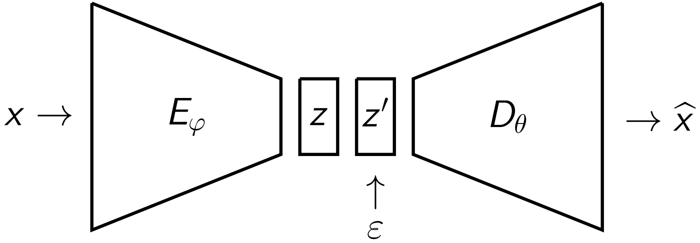
Euclidean distance $\neq$ perceptual distance

Overcoming the Limitations

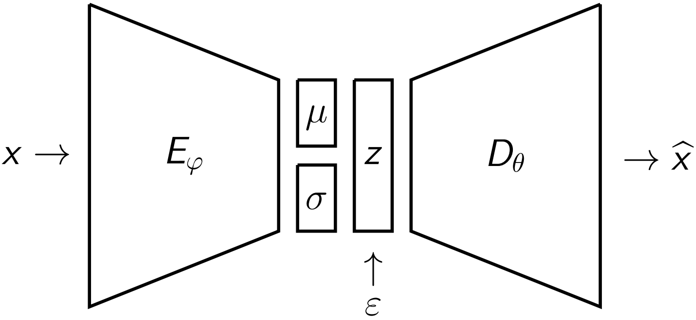
Decrease the local variability of D_θ by adding random noise to x:

$$\widehat{x} = D_\theta(\underbrace{\overbrace{E_\varphi(x)}^{z} + \varepsilon}_{z'})$$

The mapping $x \mapsto \widehat{x}$ is now *stochastic*.



$$\varepsilon \sim N(0, I)$$



$$z|x = \mu + \sigma \odot \varepsilon \sim N(\mu, \sigma\sigma^T)$$

Latent Variable Models

We want to view θ and φ as

Latent variable model:

- $z \sim p(z), \quad x|z \sim p(x|z;\theta)$
- z is a latent/hidden/explanatory variable; x is observed

This is a *generative model*. To produce an element of $\mathcal{X}$,

- sample z from $p(z)$,
- sample x from $p(x|z;\theta)$.

Want: Estimates of θ and $p(z|x;\theta)$

Autoencoders and latent variable models

$$\text{apply encoder } E_\varphi \text{ to } x \;\longleftrightarrow\; \text{sample from } p(z|x)$$

$$\text{apply dencoder } D_\theta \text{ to } z \;\longleftrightarrow\; \text{sample from } p(x|z)$$

Variational Bayes

Set up an optimization problem to give us the maximum likelihood estimator of θ.

Maximize log-likelihood, $p(x|\theta)$

Bayes theorem gives:

$$\log p(x) = \mathbb{E}_{z \sim p(z|x)} \log p(x|z) - D[p(z|x)||p(z)]$$

Approximate posterior $p(z|x)$ by $q(z|x)$:

$$\log p(x) - D[q(z|x)||p(z|x)] = \mathbb{E}_{z \sim q(z|x)} \log p(x|z) - D[q(z|x)||p(z)]$$

Instead of maximizing $p(x)$, maximize the *lower bound*:

$$\max_q \left(\mathbb{E}_{z \sim q(z|x)} \log p(x|z) - D[q(z|x)||p(z)] \right)$$

$$\max_q \left(\mathbb{E}_{z\sim q(z|x)} \log p(x|z) - D[q(z|x)||p(z)]\right)$$

Use the Gaussian family for q:

$$q(z|x;\varphi) = N(z|\mu(x|\varphi), \sigma(x|\varphi)^2))$$

Optimization problem becomes:

$$\max_q \left(\mathbb{E}_{\varepsilon\sim N(0,I)} \log p(x|z = \mu(x) + \sigma(x) \odot \varepsilon) - D[q(z|x)||p(z)]\right)$$

This can be maximized by gradient descent.