COMP 451 – Fundamentals of Machine Learning Lecture 25 — Autoencoders and self-supervision

William L. Hamilton

Autoencoders and self-supervision

- Two approaches to dimensionality reduction using deep learning.
  - This is a rough categorization and not a strict division!!

- Autoencoders:
  - Optimize a “reconstruction loss.”
  - Encoder maps input to a low-dimensional space and decoder tries to recover the original data from the low-dimensional space.

- “Self-supervision”:
  - Try to predict some parts of the input from other parts of the input.
  - I.e., make up labels from $\mathbf{x}$.

Autoencoders and self-supervision

- Two approaches to dimensionality reduction using deep learning.
  - This is a rough categorization and not a strict division!!

- Autoencoders:
  - Optimize a "reconstruction loss."
  - Encoder maps input to a low-dimensional space and decoder tries to recover the original data from the low-dimensional space.

- "Self-supervision":
  - Try to predict some parts of the input from other parts of the input.
  - I.e., make up labels from $\mathbf{x}$.

Autoencoding: the basic idea

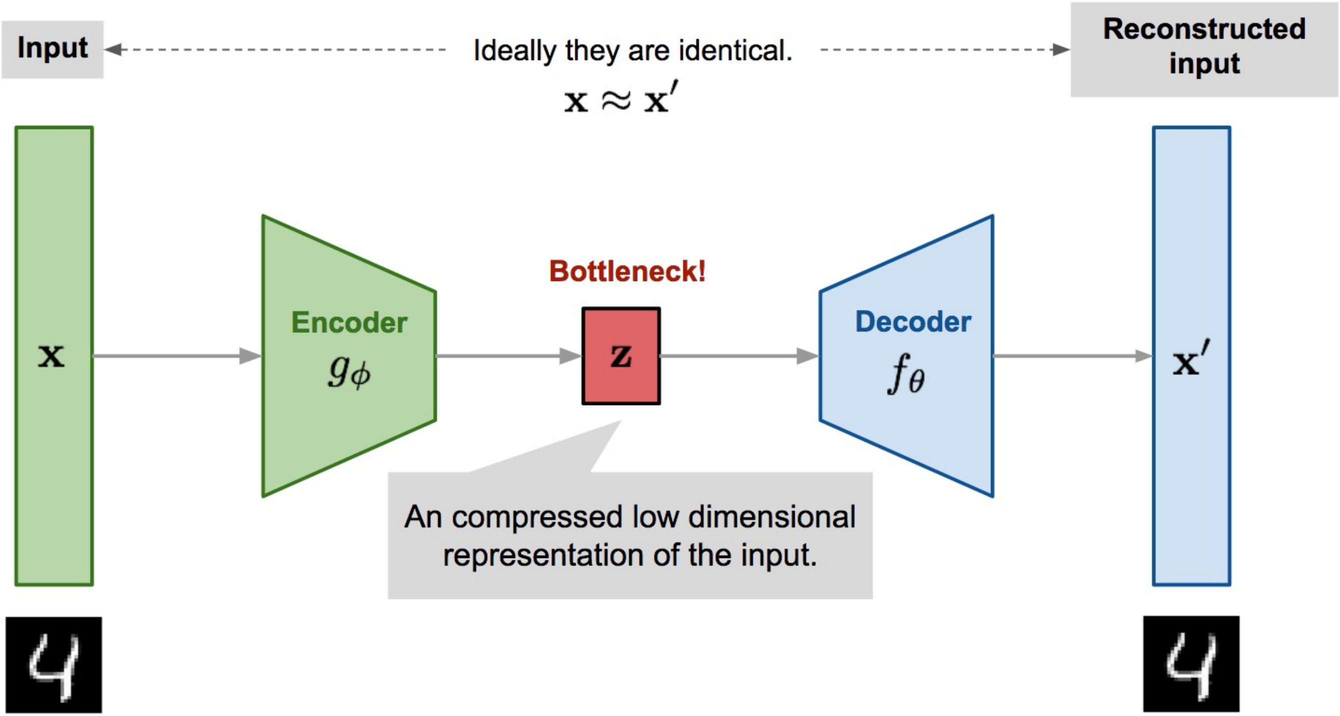


Image credit: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Learning an autoencoder function

- Goal: Learn a compressed representation of the input data.
- We have two functions (usually neural networks):
  - Encoder:

$$\mathbf{z} = g_\phi(\mathbf{x})$$

  - Decoder:

$$\hat{\mathbf{x}} = f_\theta(\mathbf{z})$$

Learning an autoencoder function

- Goal: Learn a compressed representation of the input data.
- We have two functions (usually neural networks):
  - Encoder:

$$\mathbf{z} = g_\phi(\mathbf{x})$$

  - Decoder:

$$\hat{\mathbf{x}} = f_\theta(\mathbf{z})$$

- Train using a reconstruction loss:

$$\begin{aligned} J(\mathbf{x}, \hat{\mathbf{x}}) &= \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \\ &= \|\mathbf{x} - f_\theta(g_\phi(\mathbf{x}))\|^2 \end{aligned}$$

Learning an autoencoder function

- Goal: Learn a compressed representation of the input data.
- We have two functions (usually neural networks):
  - Encoder:

$$\mathbf{z} = g_\phi(\mathbf{x})$$

  - Decoder:

$$\hat{\mathbf{x}} = f_\theta(\mathbf{z})$$

Only interesting when $\mathbf{z}$ has much smaller dimension than $\mathbf{x}$!

- Train using a reconstruction loss:

$$\begin{aligned} J(\mathbf{x}, \hat{\mathbf{x}}) &= \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \\ &= \|\mathbf{x} - f_\theta(g_\phi(\mathbf{x}))\|^2 \end{aligned}$$

Autoencoding

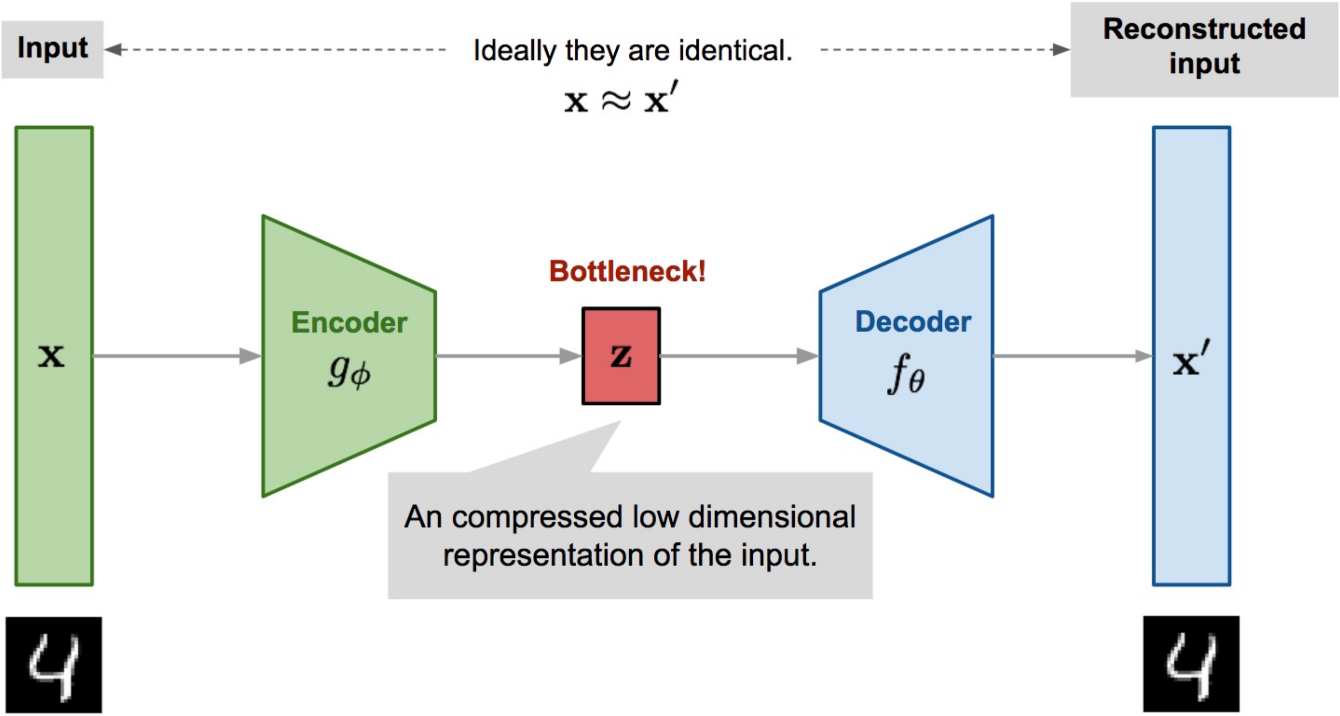


Image credit: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Recall: Principal Component Analysis (PCA)

- Idea: Project data into a lower-dimensional sub-space, $R^m \dashrightarrow R^{m'}$, where $m' < m$.

Recall: Principal Component Analysis (PCA)

- **Idea:** Project data into a lower-dimensional sub-space, $R^m \dashrightarrow R^{m'}$, where $m' < m$.

- Consider a linear mapping, $x_i \dashrightarrow W^T x_i$
  - W is the compression matrix with dimension $R^{m \times m'}$.
  - Assume there is a decompression matrix $U^{m' \times m}$.

Recall: Principal Component Analysis (PCA)

- Idea: Project data into a lower-dimensional sub-space, $R^m \dashrightarrow R^{m'}$, where $m' < m$.

- Consider a linear mapping, $x_i \dashrightarrow W^T x_i$
  - W is the compression matrix with dimension $R^{m x m'}$.
  - Assume there is a decompression matrix $U^{m' x m}$.

- Solve the following problem: $\text{argmin}_{W,U} \Sigma_{i=1:n} || x_i - UW^T x_i ||^2$

Recall: Principal Component Analysis (PCA)

- Solve the following problem: $\text{argmin}_{W,U} \Sigma_{i=1:n} || x_i - UW^T x_i ||^2$
- Equivalently: $\text{argmin}_{W,U} || X - XWU^T ||^2$

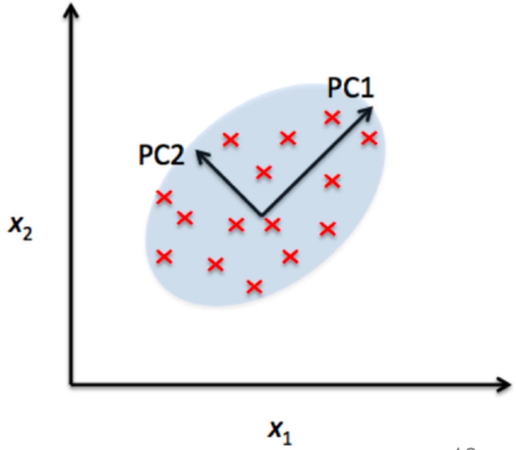
- Solution is given by eigen-decomposition of X^TX.
  - W is mxm' matrix corresponding to the first m' eigenvectors of X^TX (sorted in descending order by the magnitude of the eigenvalue).
  - Equivalently: W is mxm' matrix containing the first m' left singular vectors of X
  - **Note:** The columns of W are orthogonal!

PCA vs autoencoders

In the case of a linear encoders and decoders:

$$f_W(x) = Wx \qquad\qquad g_{\hat{W}}(h) = W'h\,,$$

with squared-error reconstruction loss we can show that the minimum error solution W yields the same subspace as PCA.

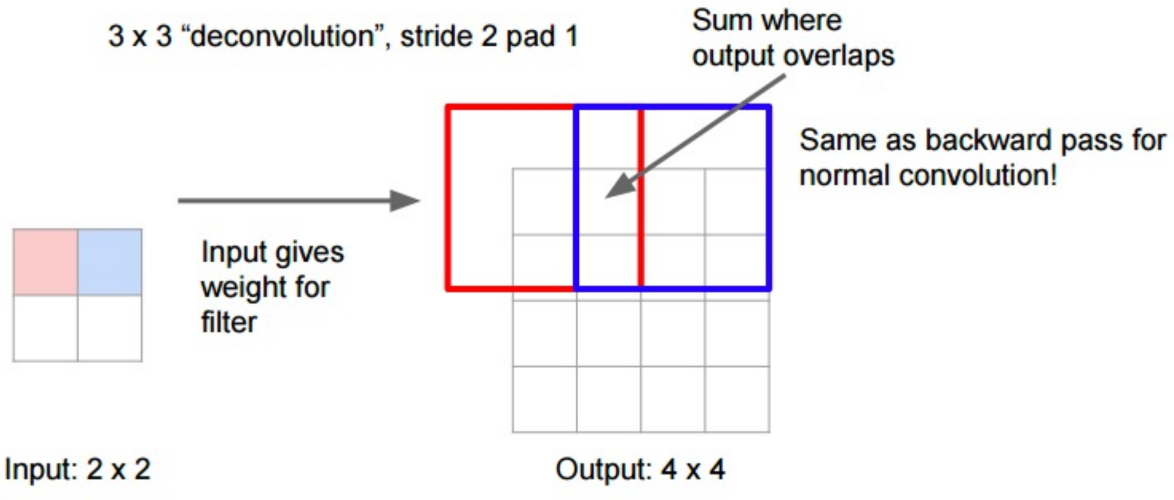


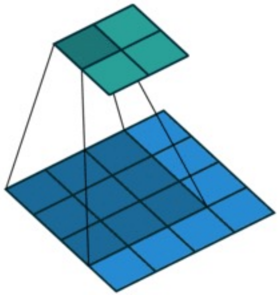
More advanced encoders and decoders

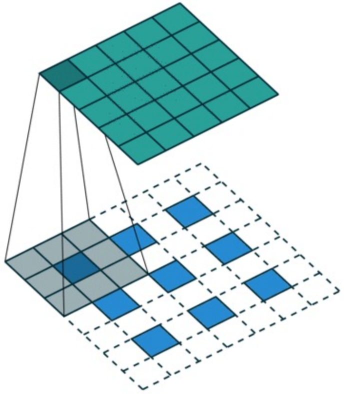
- What to use as encoders and decoders?
- Most data (e.g., arbitrary real-valued or categorical features).
  - Encoder and decoder are feed-forward neural networks.
- Sequence data
  - Encoder and decoder are RNNs.
- Image data
  - Encoder is a CNN; decoder is a deconvolutional network.

Aside: Deconvolutions

- "Deconvolution" is just a transposed convolution.







Regularization of autoencoders

- How can we generate sparse autoencoders? (And also, why?)

Regularization of autoencoders

- How can we generate sparse autoencoders? (And also, why?)

- Weight tying of the encoder and decoder weights $(\theta = \phi)$ to explicitly constrain (regularize) the learned function.

Regularization of autoencoders

- How can we generate **sparse** autoencoders? (And also, why?)

- Weight tying of the encoder and decoder weights $(\theta = \phi)$ to explicitly constrain (regularize) the learned function.

- Directly penalize the output of the hidden units (e.g. with L1 penalty) to introduce sparsity in the weights.

Regularization of autoencoders

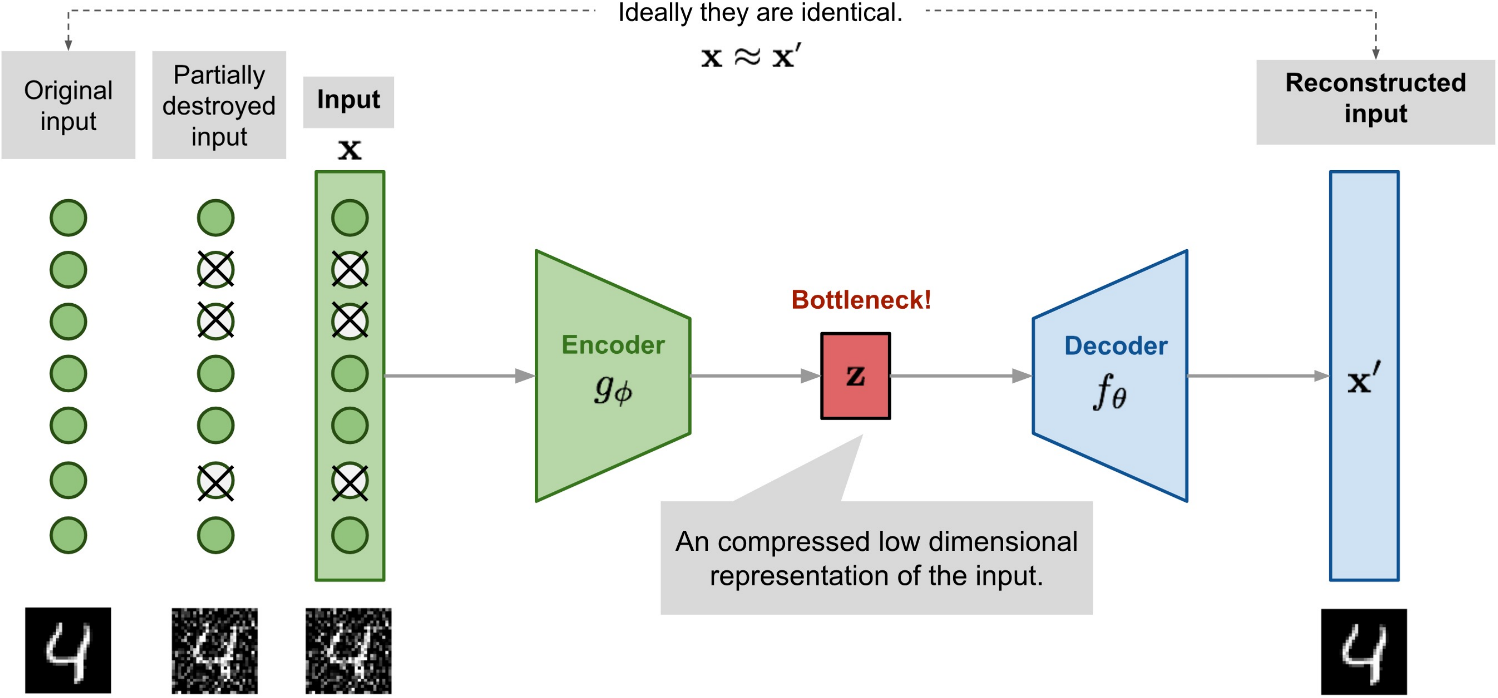
- How can we generate sparse autoencoders? (And also, why?)

- Weight tying of the encoder and decoder weights $(\theta = \phi)$ to explicitly constrain (regularize) the learned function.

- Directly penalize the output of the hidden units (e.g. with L1 penalty) to introduce sparsity in the weights.

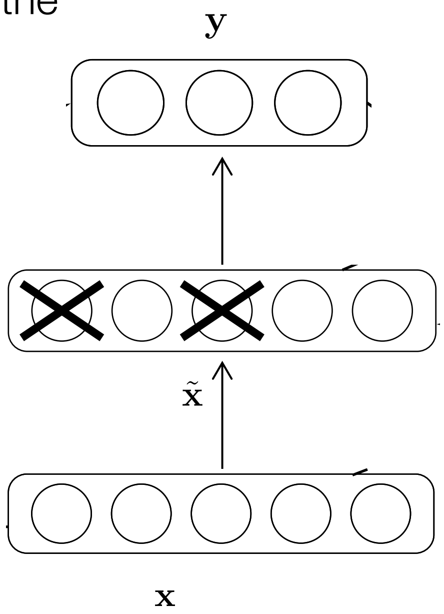
- Penalize the average output (over a batch of data) to encourage it to approach a fixed target.

Denoising autoencoders



Denoising autoencoders

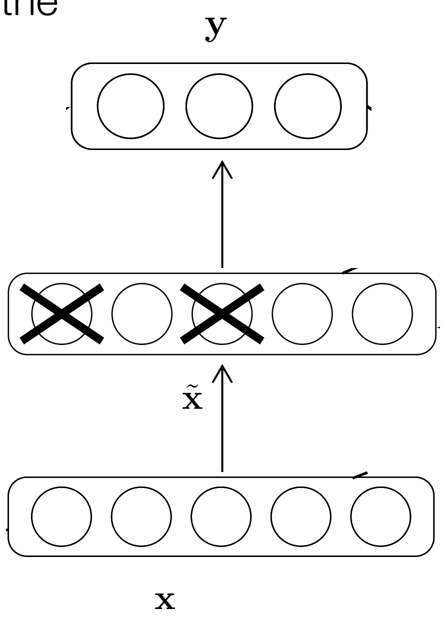
- Idea: To force the hidden layer to discover more robust features, train the autoencoder with a corrupted version of the input.



Denoising autoencoders

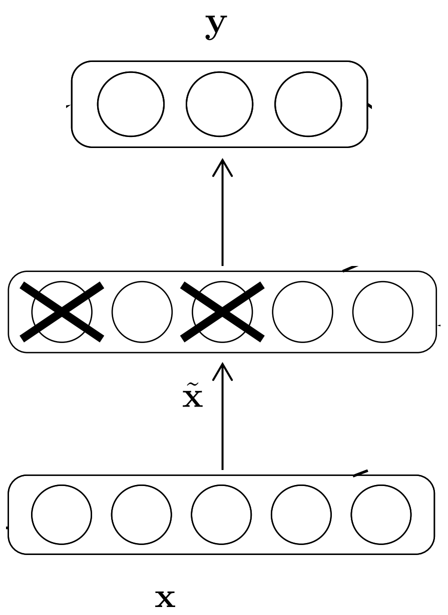
- Idea: To force the hidden layer to discover more robust features, train the autoencoder with a corrupted version of the input.

- Corruption processes:
  - Additive Gaussian noise
  - Randomly set some input features to zero.
  - *More noise models in the literature*.



Denoising autoencoders

- **Idea**: To force the hidden layer to discover more robust features, train the autoencoder with a corrupted version of the input.



- **Corruption processes**:
  - Additive Gaussian noise
  - Randomly set some input features to zero.
  - *More noise models in the literature*.

- **Training criterion**:

$$\text{Err} = \sum_{i=1:n} E_{q(xi'|xi)}\, L\, [\, x_i\, ,\, f_{W'}\, (g_W(x_i'))\,]$$

where L is some reconstruction loss x is the original input, x' is the corrupted input, and $q()$ is the corruption process.

Contractive autoencoders

- Goal: Learn a representation that is robust to noise and perturbations of the input data, by regularizing the latent space (represented by L2 norm of the Jacobian of the encoded input.)

- Contractive autoencoder training criterion:

$$Err(W,W') = \sum_{i=1:n} L\,[\,x_i\,,\,f_{W'}\,(g_W(x_i))\,] + \lambda||J(x_i)||_F^2$$

where L is some reconstruction loss, $J(x_i)=\partial f_W(x_i)/\partial x_i$ is a Jacobian matrix of the encoder evaluated at x_i, F is the Frobenius norm, and λ controls the strength of the regularization.

Many more similar ideas in the literature…

Autoencoding: The key idea

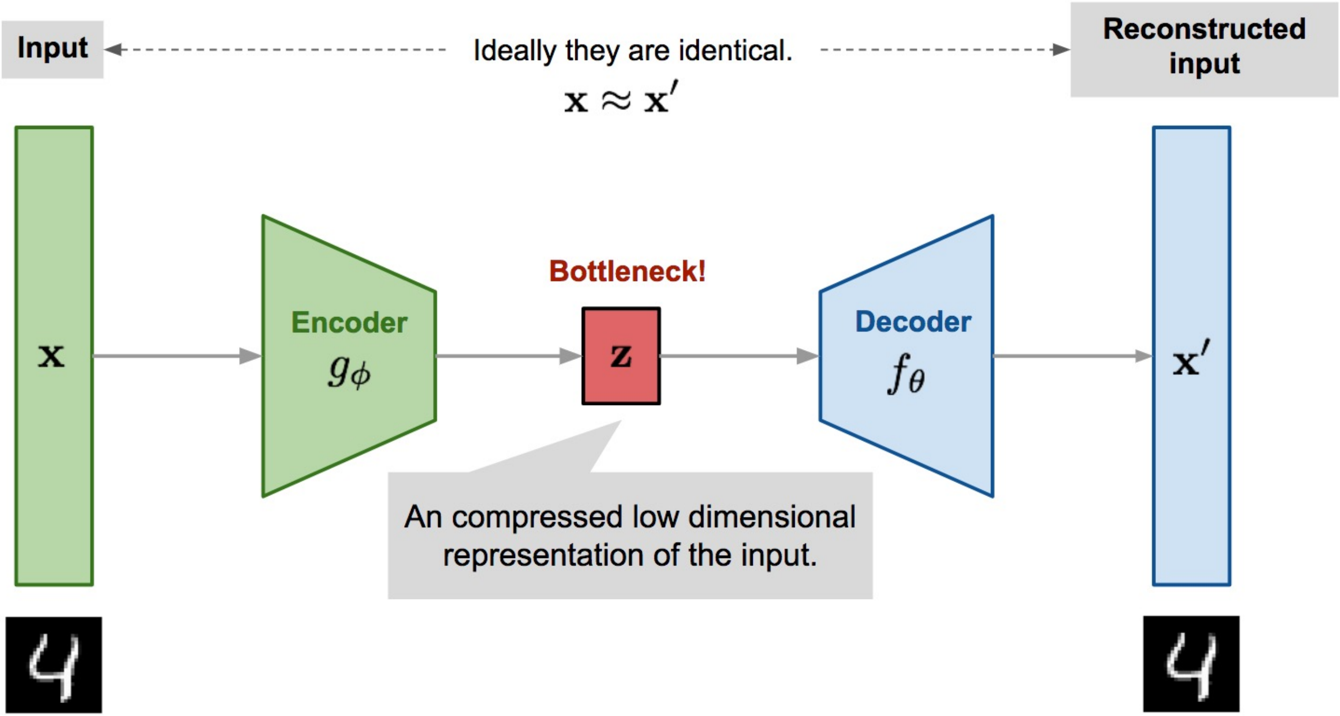
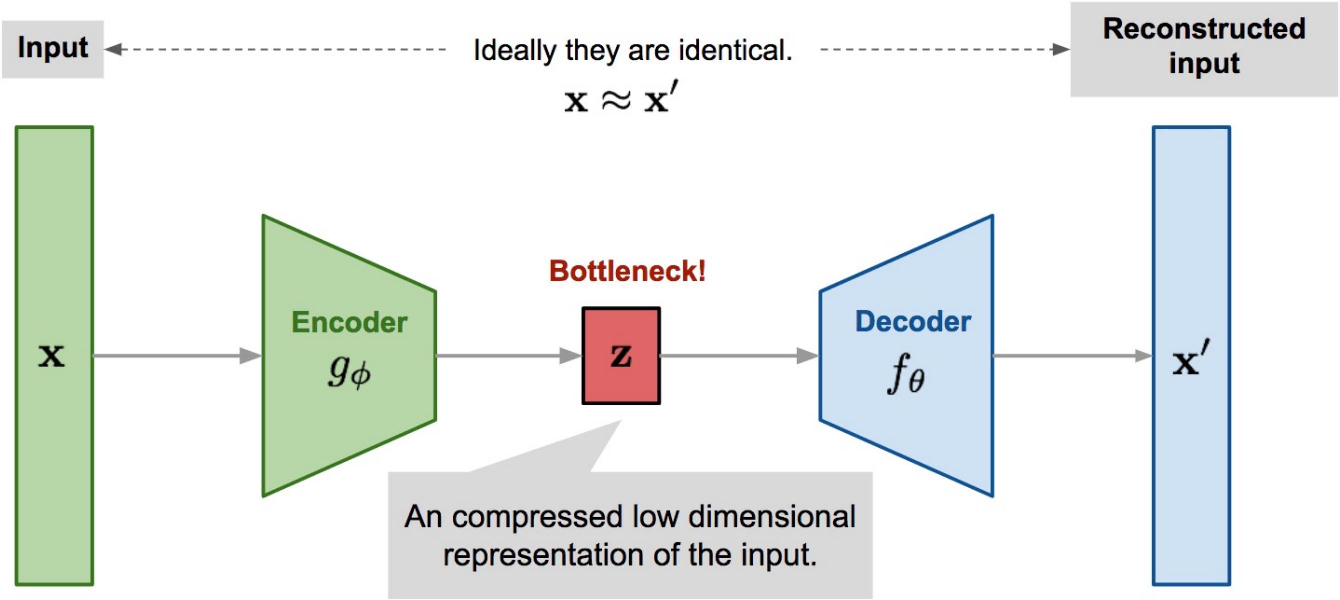


Image credit: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Autoencoding language?



"dog" "dog"

Autoencoding language

- Autoencoding can generate high-quality low-dimensional features.

- Works well when the original space $\mathbf{x}$ is rich and high-dimensional.
  - Images
  - MRI data
  - Speech data
  - Video

- But what if we don't have a good feature space to start with? E.g., for representing words?

Autoencoders and self-supervision

- Two approaches to dimensionality reduction using deep learning.
  - This is a rough categorization and not a strict division!!

- Autoencoders:
  - Optimize a "reconstruction loss."
  - Encoder maps input to a low-dimensional space and decoder tries to recover the original data from the low-dimensional space.

- "Self-supervision":
  - Try to predict some parts of the input from other parts of the input.
  - I.e., make up labels from $\mathbf{x}$.

Words embeddings / self-supervision

- What is the input space for words/language?
- In the unsupervised case, generally all we have is a text corpus (i.e., a set of documents).
- How can we learn representations from this data?

Words embeddings / self-supervision

- Idea: Make up a supervised learning task in order to learn representations for words!

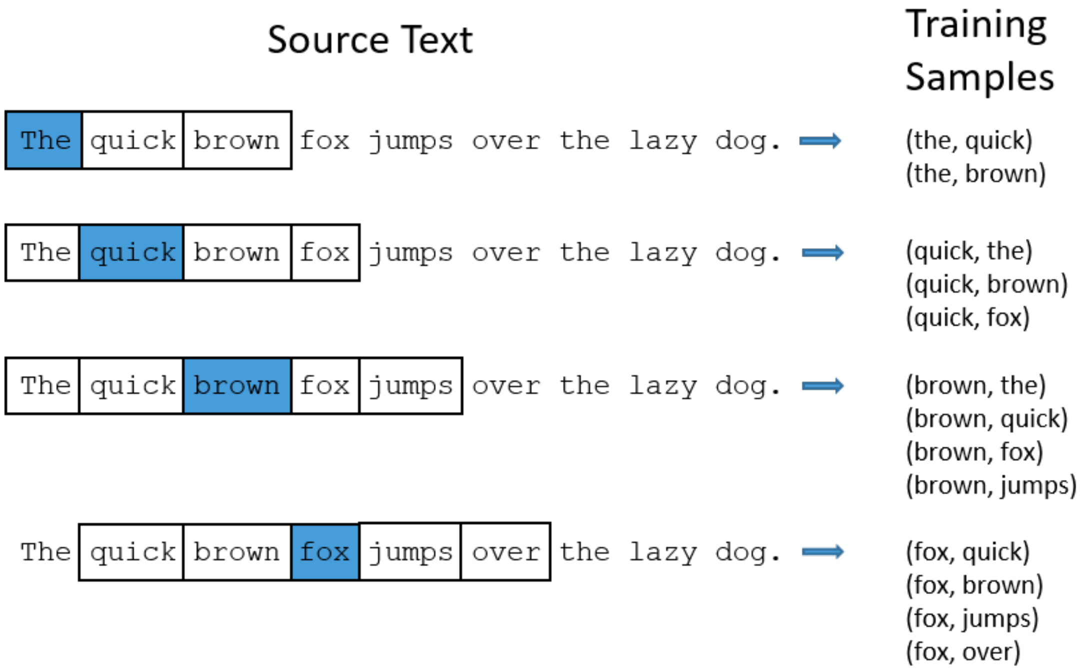
- Co-occurrence information tells us a lot about word meaning.
  - For example, “dog” and “pitbull” occur in many of the same contexts.
  - For example, “loved” and “appreciated” occur in many of the same contexts.

- Let’s learn features/representations for words that are good for predicting their context!

Words embeddings / self-supervision

- Create a training set by applying a “sliding window” over the corpus.

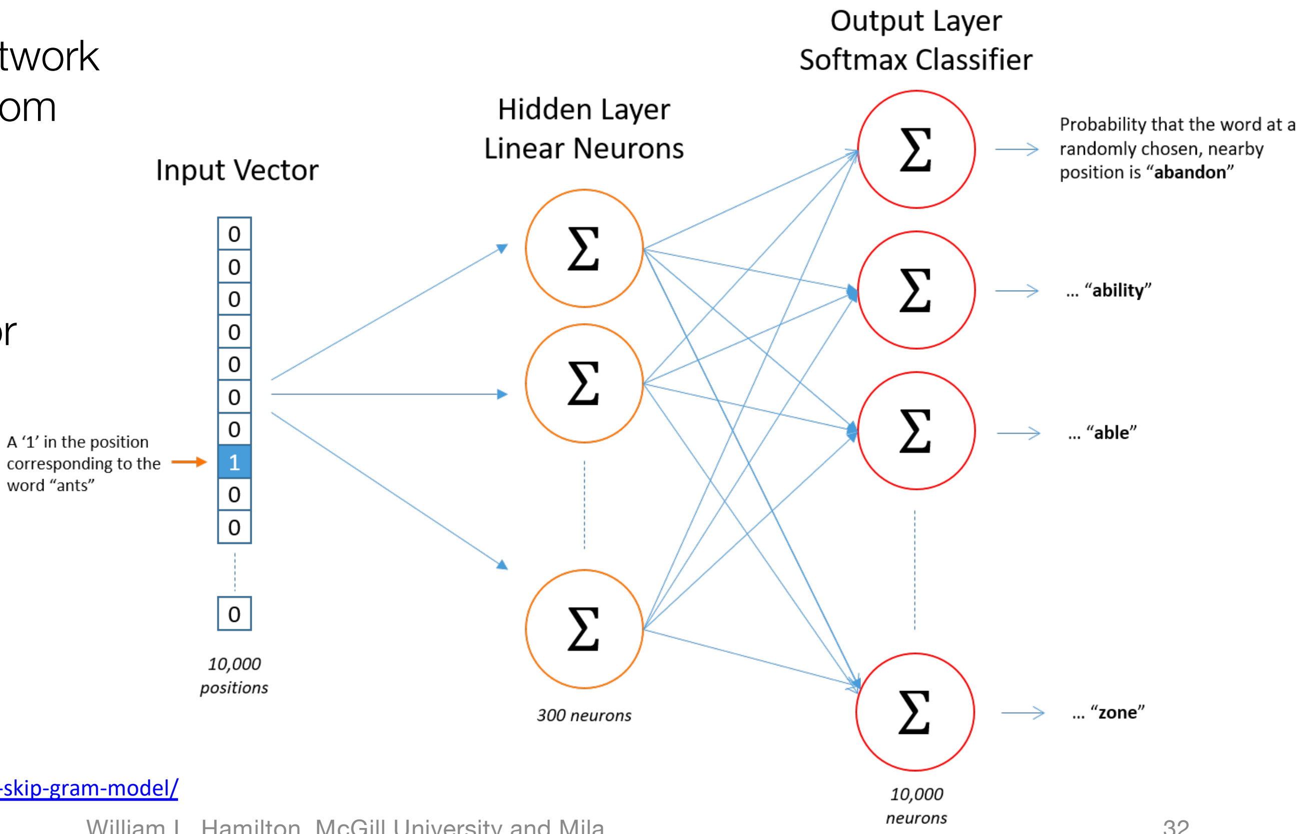
- I.e., given a word, we try to predict what words are likely to occur around it.



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec / SkipGram Model

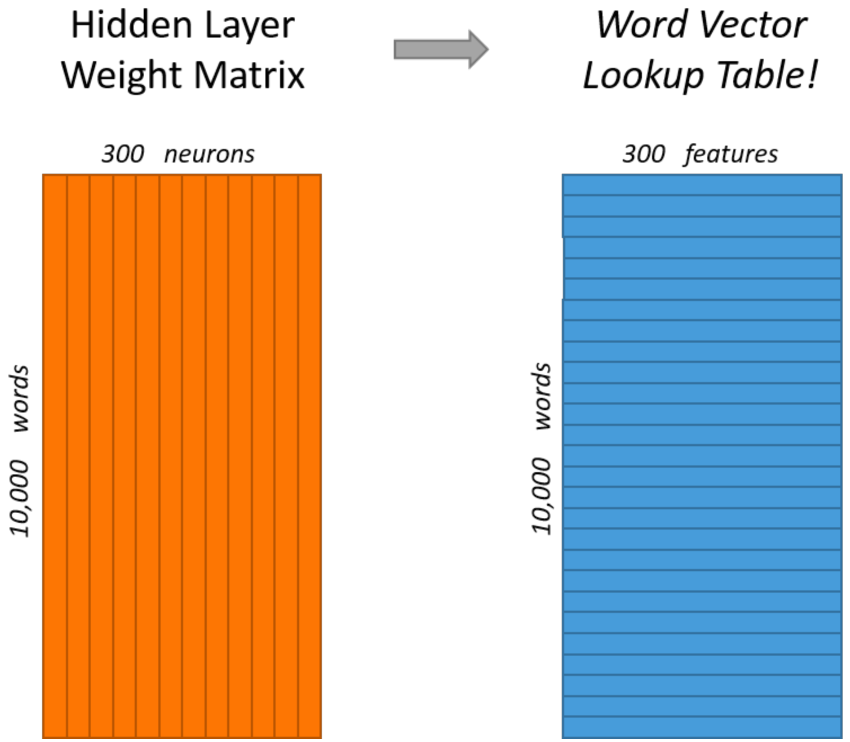
- Key idea: Train a neural network to predict context words from input word.
- Hidden layer learns representations/features for words!



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec / SkipGram Model

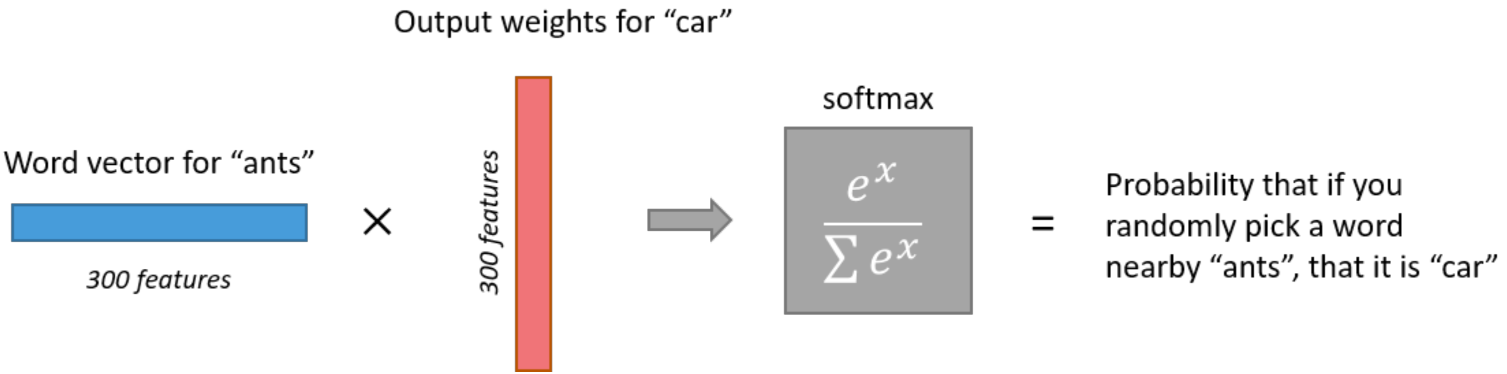
- Key idea: Train a neural network to predict context words from input word.
- Hidden layer learns representations/features for words!



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word2Vec / SkipGram Model

- Intuition: dot-product between word representations is proportional to the probability that they co-occur in the corpus!
- One issue is that the output layer is very big!
  - We need to do a softmax over the entire vocabulary!



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Negative sampling

- Instead of using a softmax, we approximate it!

- Original softmax loss:

Sum over entire vocabulary

Dot-product of word embeddings

$$\boxed{-\log(P(w,c))} = -\log\left(\frac{e^{\boxed{\mathbf{z}_w^\top \mathbf{z}_c}}}{\sum_{\boxed{c' \in \mathcal{V}}} e^{\mathbf{z}_w^\top \mathbf{z}_{c'}}}\right)$$

$$= -\mathbf{z}_w^\top \mathbf{z}_c + \boxed{\log\left(\sum_{c' \in \mathcal{V}} e^{\mathbf{z}_w^\top \mathbf{z}_{c'}}\right)}$$

Negative log-likelihood of seeing word *c* in the context of word *w*

This term is very expensive! O(|V|)

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Negative sampling

- Instead of using a softmax, we approximate it!
- Negative sampling loss:

Instead of summing over entire vocabulary. We just sample N "negative example" words.

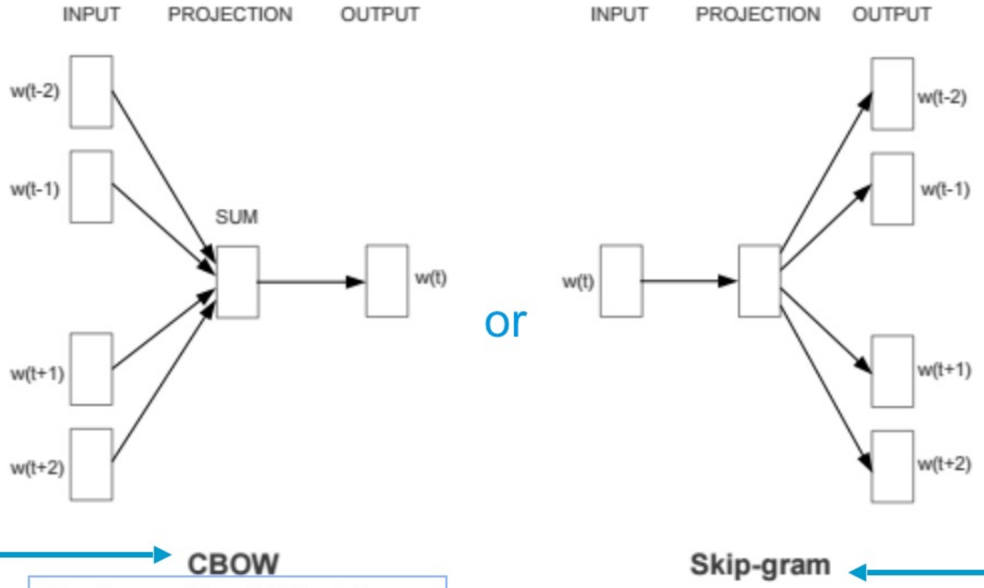
$$-\log(P(w,c)) \approx \boxed{-\log(\sigma(\mathbf{z}_w^\top \mathbf{z}_c))} - \boxed{\sum_{j=1}^{N} \log(\sigma(-\mathbf{z}_w^\top \mathbf{z}_{c'_j}))}$$

Probability that w and c co-occur approximated with sigmoid.

Key idea: Dot-product for co-occurring pairs should be higher than random pairs!.

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Variants of word2vec



or

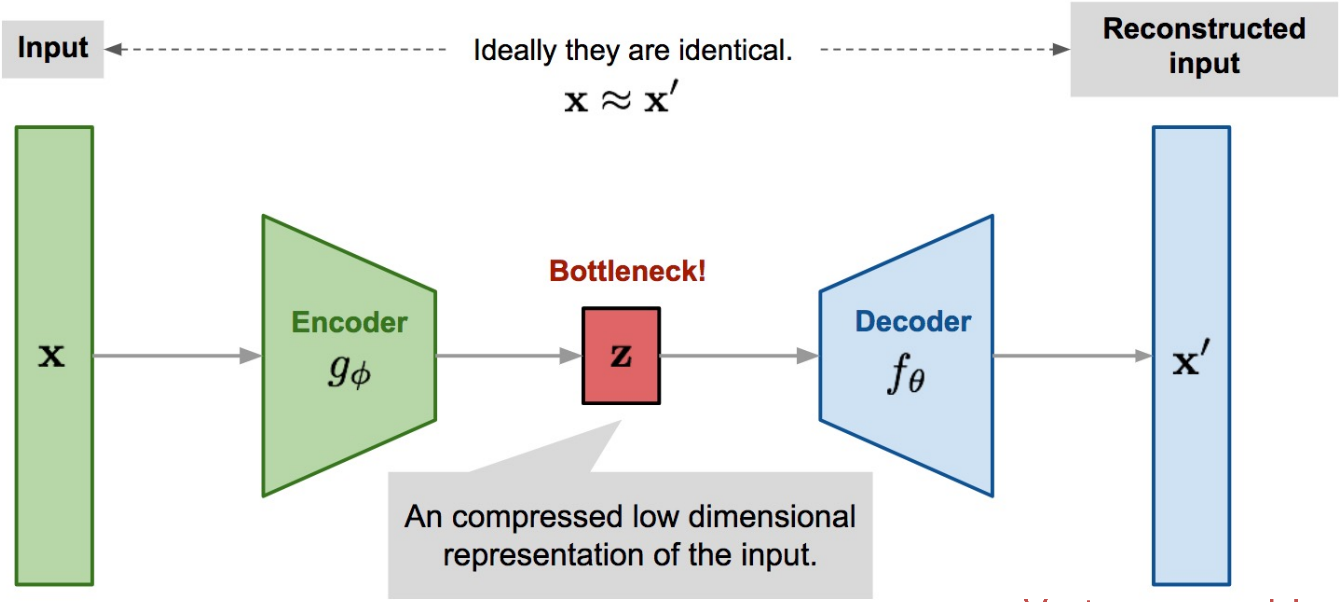
Given a set of (neighboring) words, **guess single words** that potentially occur along with this set of words.

(Continuous Bag Of Words)

Guess potential neighboring words based on the single word being analyzed.

https://towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281

Word2Vec and autoencoders



Vector summarizing a word's co-occurrence statistics.

Vector summarizing a word's co-occurrence statistics.

"Self-supervised learning" more generally

- Key idea: Create supervised data from unsupervised data by predicting some parts of the input from other parts of the input.
- A relatively new/recent idea.
- Has led to new state-of-the-art in language and vision tasks.
- E.g., BERT and ELMO in NLP.