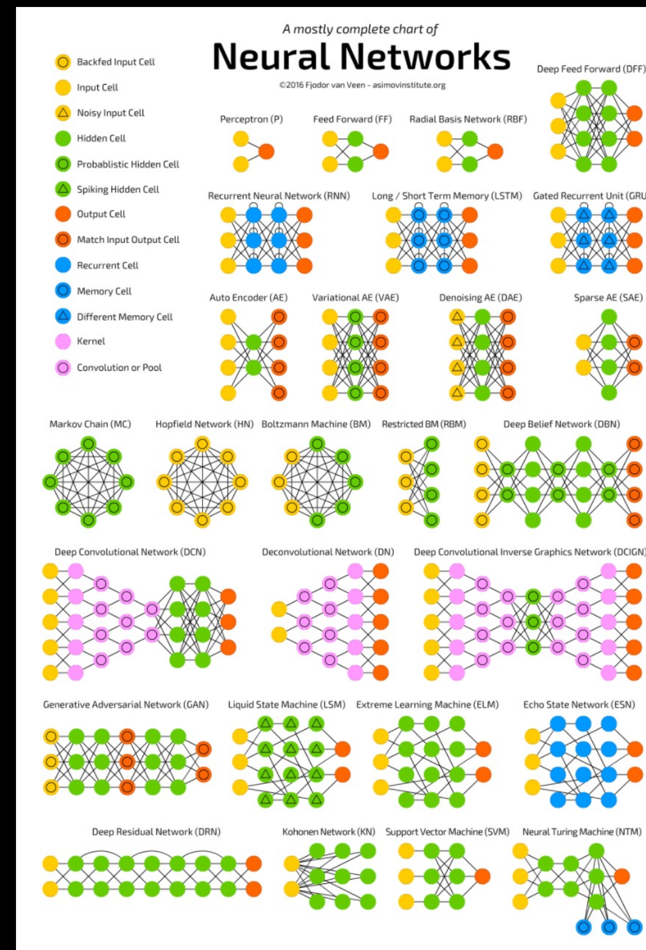
Autoencoders

CSCI 4360/6360 Data Science II

The Neural Network Zoo

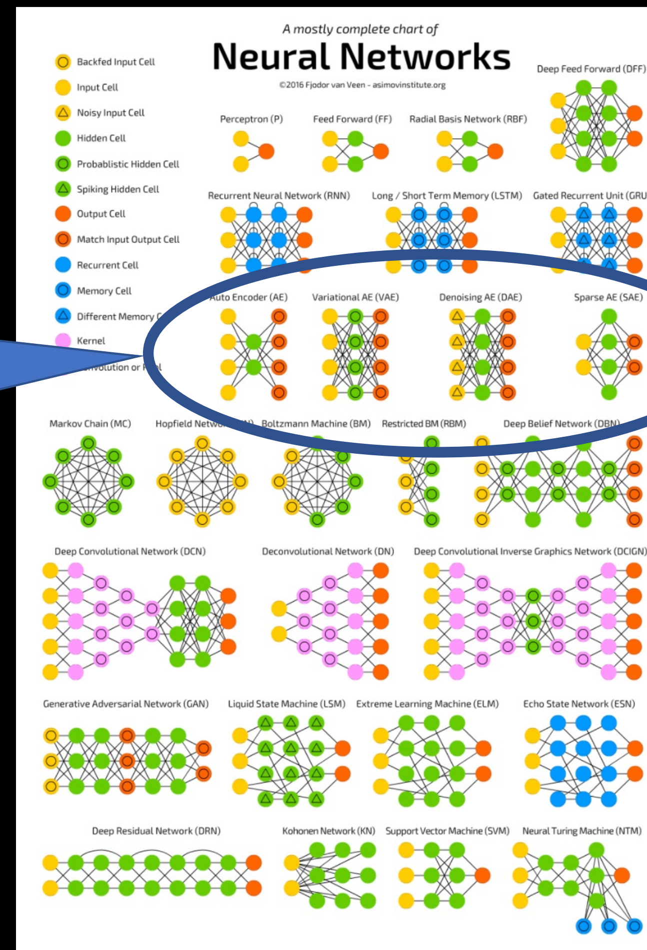
- http://www.asimovinstitute.org/neural-network-zoo/



The Neural Network Zoo

- http://www.asimovinstitute.org/neural-network-zoo/

Today

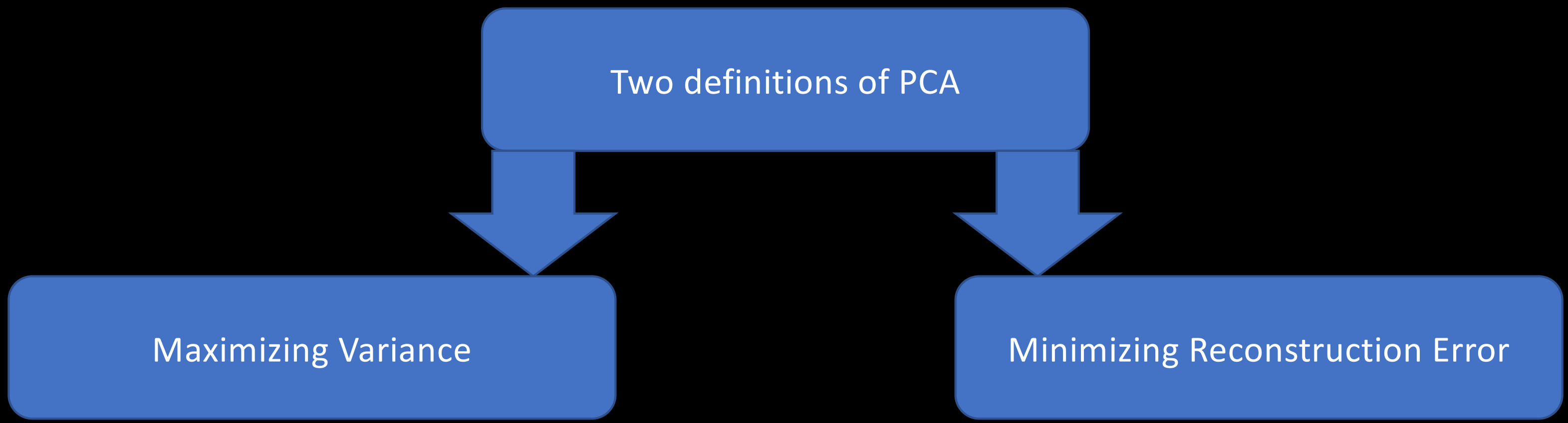


Dimensionality Reduction

- Reduce the number of random variables under consideration
  - Reduce computational cost of downstream analysis
  - Remove sources of noise in the data
  - Define an embedding of the data
  - Elucidate the manifold of the data
- **We've covered several strategies so far**

Principal Component Analysis (PCA)

1. Orthogonal projection of data
2. Lower-dimensional linear space known as the *principal subspace*
3. Variance of the projected data is maximized



Kernel PCA

- In kernel PCA, we consider data that have already undergone a nonlinear transformation:

$$\vec{x} \in \mathcal{R}^D \quad \Rightarrow \quad \phi(\vec{x}) \in \mathcal{R}^M$$

- **We now perform PCA on this new *M*-dimensional feature space**

Sparse PCA

- We still want to maximize $u_i^T S u_i$, subject to $u_i^T u_i = 1$
- …and one more constraint: we want to *minimize* $||u_i||_1$

- Formalize these constraints using Lagrangian multipliers

$$\min_{W,U} ||X - WU^T||_F^2 + \gamma \sum_{n=1}^{N} ||\vec{w}_i||_1 + \gamma \sum_{i=1}^{D} ||\vec{u}_i||_1$$

Stochastic SVD (SSVD)

- Uses **random projections** to find close approximation to SVD
- Combination of probabilistic strategies to maximize convergence likelihood
- Easily scalable to *massive* linear systems

Dictionary Learning

- This gives the minimization

$$\min_{B,\Theta} \sum_{i=1}^{n} \left(||\vec{x}_i - B\vec{\theta}_i||_q^q + h(\vec{\theta}_i) \right)$$

  where *h* promotes sparsity in the coefficients, and *B* is chosen from a constraint set

- The general dictionary learning problem then follows

$$\phi(\Theta, B) = \frac{1}{2}||X - B\Theta||_F^2 + h(\Theta) + g(B)$$

  where specific choices of *h* and *g* are what differentiate the different kinds of dictionary learning (e.g. hierarchical, K-SVD, etc)

Autoencoders

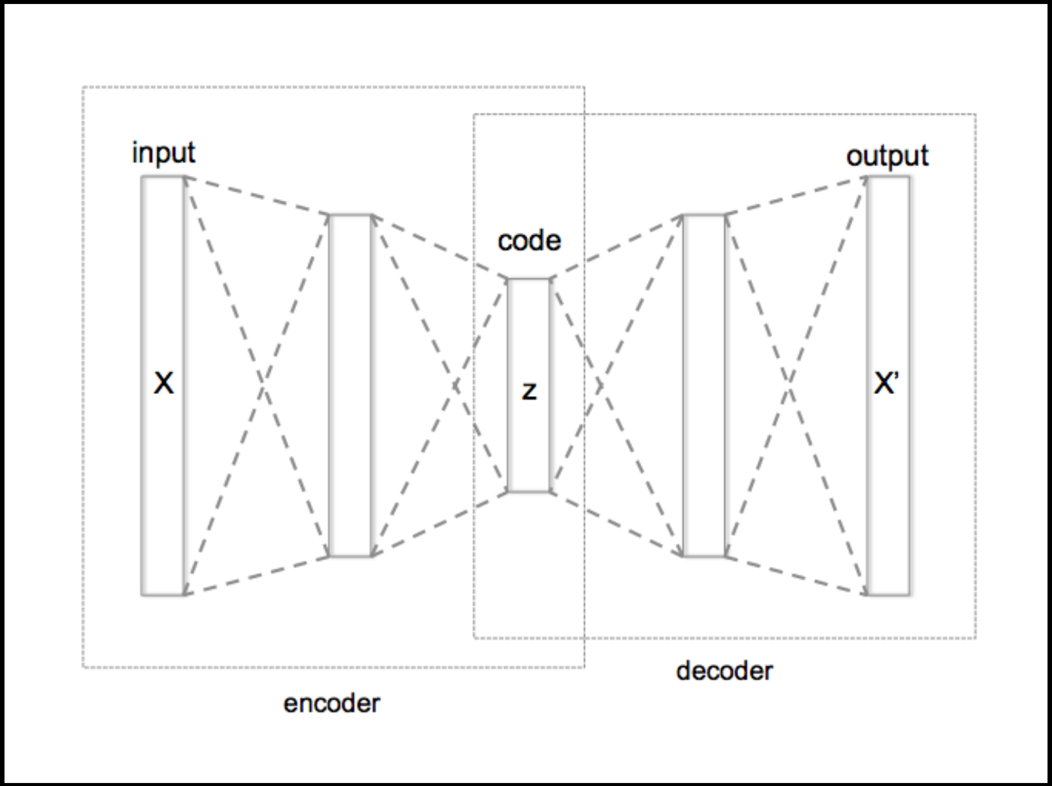
- "Self encode"
- ANNs with output = input

$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$

$$\psi : \mathcal{F} \rightarrow \mathcal{X}$$

$$\phi, \psi = \arg\min_{\phi,\psi} ||X - (\psi \circ \phi)X||^2$$

- Identical to the LSTM's encoder-decoder architecture



Autoencoders

- Learn a "non-trivial" identity function
- Low-dimensional "code"

- **No other assumptions**

+

- Very compact representation
- No strong *a priori* form (flexible)

-

- Difficult to interpret
- Prone to "collapse"

- PCA: maximize variance / minimize reconstruction
  - Linearly independent
  - Gaussian
- Dictionary Learning: sparse code / minimize reconstruction
  - Nonlinear
- Kernel / Sparse PCA

Autoencoders

- Key point: autoencoders should be **undercomplete**
  - Code dimension < input dimension

$$L(\vec{x}, g(f(\vec{x})))$$

- *L* is some loss function penalizing *g(f(x))* for being dissimilar from *x*
- If *f* and *g* are linear, and *L* is mean squared error, undercomplete AE learns to span the same subspace as PCA

$$\phi, \psi = \arg\min_{\phi,\psi} ||X - (\psi \circ \phi)X||^2$$

$$U = \arg\min_{U} ||X - U\Lambda U^T||^2$$

Sparse Autoencoders

- *g(h)* is decoder output
- *h = f(x)*, encoder output
- *Ω* is sparsity penalty

$$L(\vec{x}, g(f(\vec{x}))) + \Omega(\vec{h})$$

- Note on regularizer

$$p(\vec{\theta}, \vec{x}) \equiv \log p(\vec{x}|\vec{\theta}) + \log p(\vec{\theta})$$

No straightforward Bayesian interpretation of regularizer

"Typical" penalties can be viewed as a MAP approximation to Bayesian inference, with regularizers as priors over parameters

Regularized MAP then maximizes:

But autoencoder regularization relies **only** on the data. **It's more of a "preference over functions" than a prior.**

Denoising Autoencoders

- Instead of learning

$$L(\vec{x}, g(f(\vec{x})))$$

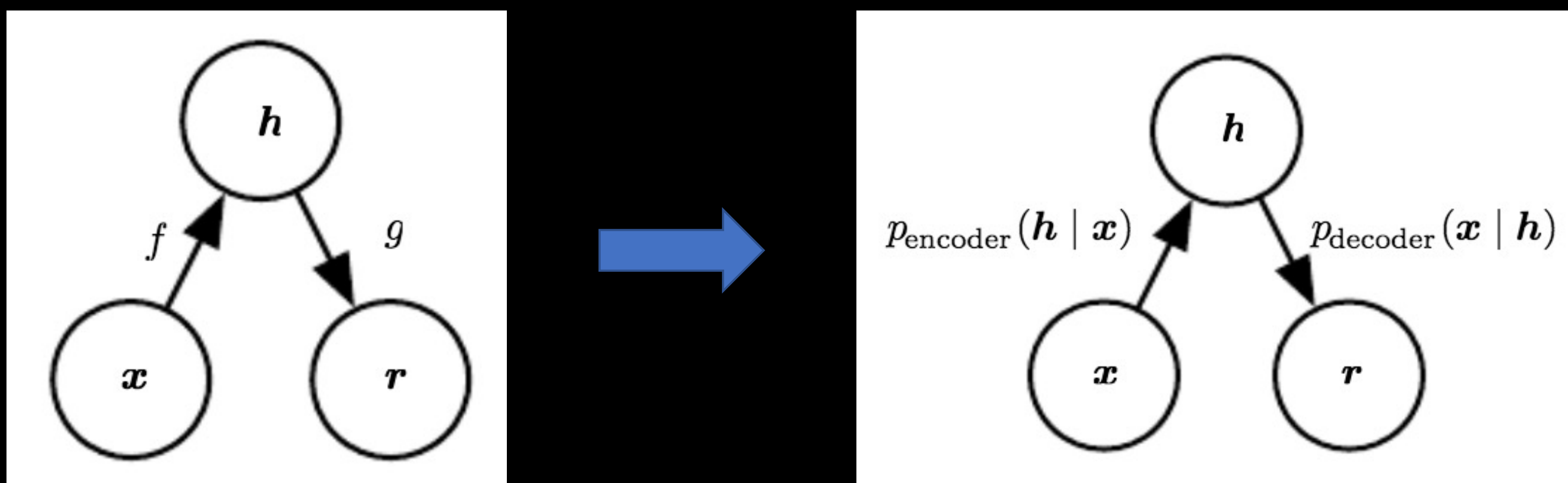
- Learn

$$L(\vec{x}, g(f(\tilde{x})))$$

  where $\tilde{x}$ is a corrupted version of x
- Forces the autoencoder to learn the structure of $p_{data}(x)$
- **Form of "stochastic encoder / decoder"**

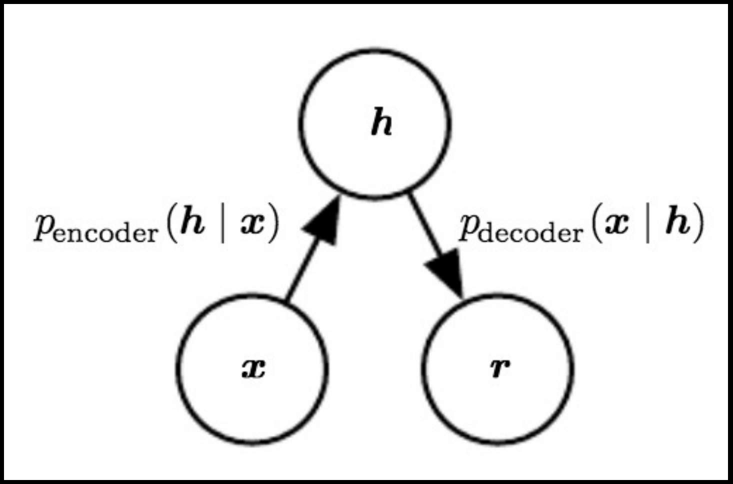
Denoising Autoencoders

- No longer deterministic!
- Given a hidden code h, minimize $-\log p_{decoder}(x|h)$



Denoising Autoencoders

- Generalize encoding function to *encoding distribution*
  $$p_{\text{encoder}}(\vec{h}|\vec{x}) = p_{\text{model}}(\vec{h}|\vec{x})$$
- Same with the *decoding distribution*
  $$p_{\text{decoder}}(\vec{x}|\vec{h}) = p_{\text{model}}(\vec{x}|\vec{h})$$
- Together, these comprise a *stochastic encoder and decoder*



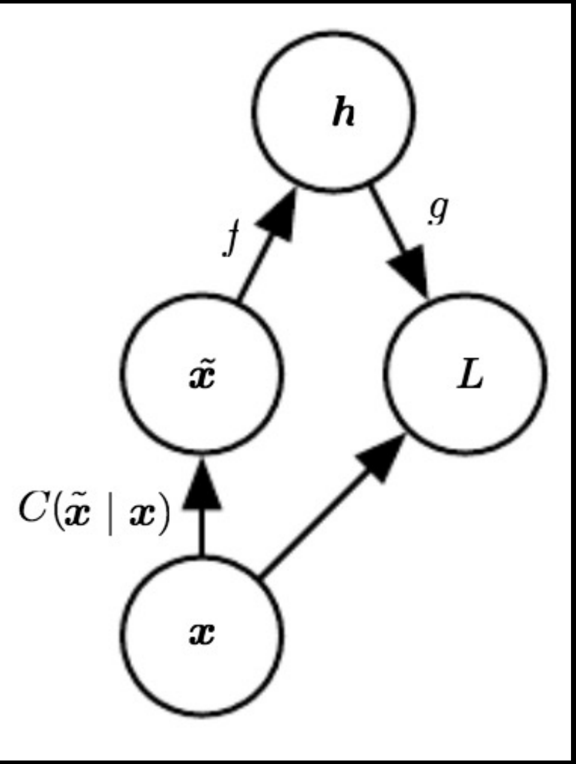
Denoising Autoencoders

- Define a corruption process, *C*

$$C(\tilde{x}|\vec{x})$$

- Autoencoder learns a *reconstruction distribution* $p_{reconstruct}(x\ |\tilde{x})$

1. Sample a training example *x*
2. Sample a corrupted version $\tilde{x}$ from *C*
3. Use $(x, \tilde{x})$ as a training pair



Denoising Autoencoders

- Optimize

$$-\mathbb{E}_{\vec{x}\sim\hat{p}_{\text{data}}}(\vec{x})\mathbb{E}_{\tilde{x}\sim C(\tilde{x}|\vec{x})}\log p_{\text{decoder}}(\vec{x}|\vec{h}=f(\tilde{x}))$$

Sample from training set and compute expectation

Expectation over corrupted examples

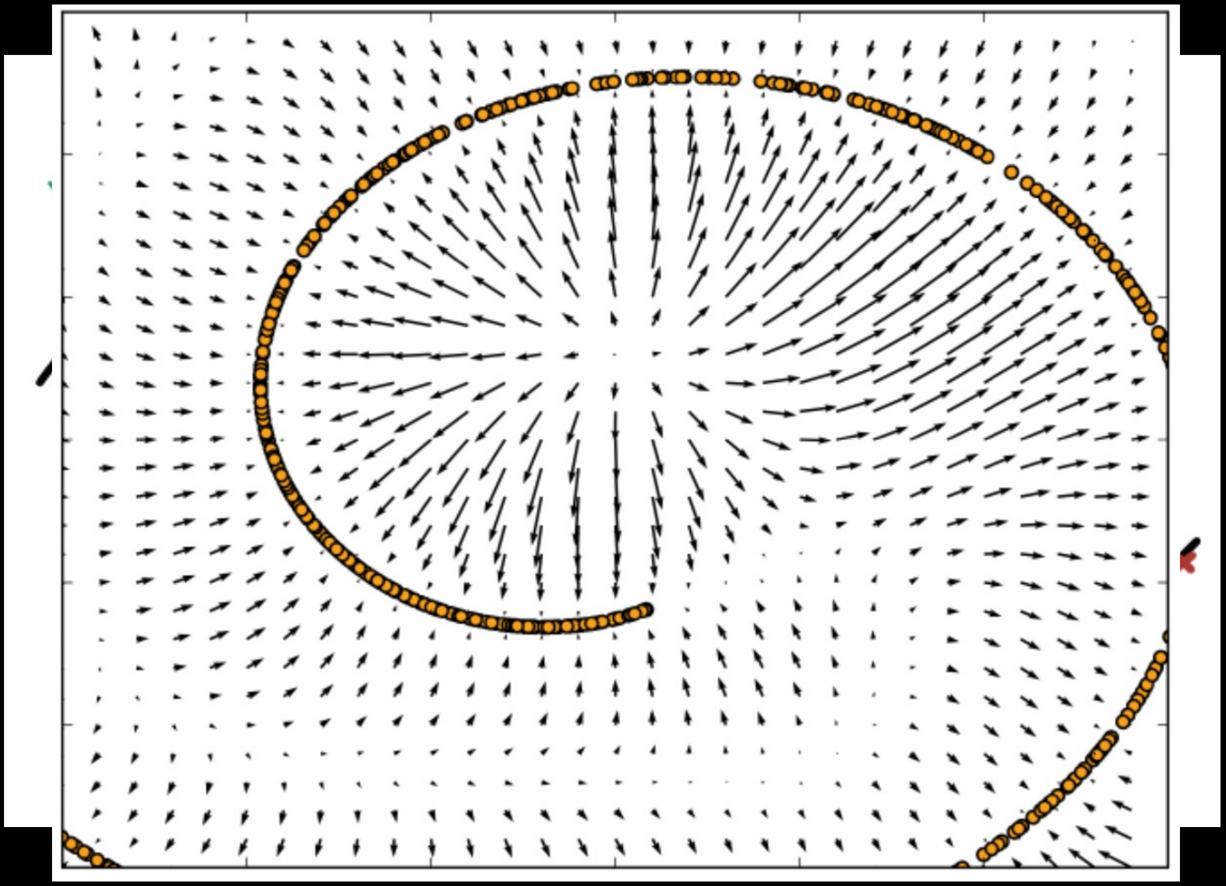
…with respect to learning the *uncorrupted data* from the encoded corrupted data

- Easy choice of *C*

$$C(\tilde{x}|\vec{x})=\mathcal{N}(\tilde{x};\mu=\vec{x},\Sigma=\sigma^2 I)$$

Denoising Autoencoders

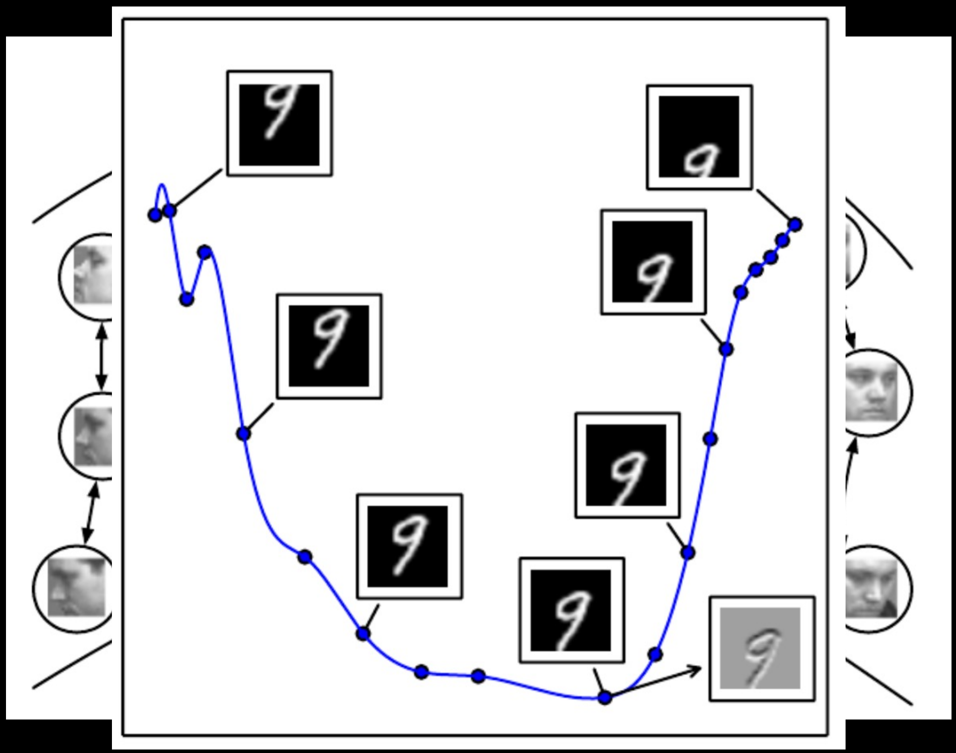
- DAEs train to map $\tilde{x}$ back to uncorrupted x
- Gray circle = equiprobable C
- Vector from $\tilde{x}$ points approximately to nearest x on manifold
- **DFA learns a vector field around a manifold**



Embeddings

- Manifolds would seem to imply *representation learning* beyond a simple low-dimensional code

- Autoencoders can learn powerful relationships in this regard
  - Pose
  - Position
  - Affine transformations



Generative Models

- Go beyond learning *x -> h*, instead focused on learning *p(x, h)*

- Manifold learning with Autoencoders
- Variational Autoencoders (VAEs)
- Deep Belief Networks (DBNs)
- Deep Restricted Boltzmann Machines (DBMs)
- Generative Adversarial Networks (GANs)

- **More next week!**

0 response submitted

If X is your data and Y are your labels, which of the following represents a generative distribution?

- P(X | Y)
- P(X)
- P(X, Y)
- None of the above
- All of the above

Treemap Bar

1 of 1

Conclusions

- Autoencoders
  - Multilayer perceptron (ANN) that is symmetric
  - Output = input
  - Goal is to learn a non-trivial identity function, or an undercomplete code *h*
- Sparse Autoencoders
  - Include a sparsity constraint on the code
- Denoising Autoencoders
  - Learn a mapping to de-corrupt data
  - Include a corruption process *C*
  - Equates to a traversal of the data manifold -> **generative modeling primer**

References

- *Deep Learning Book*, Chapter 14: “Autoencoders”
  http://www.deeplearningbook.org/contents/autoencoders.html
- DL4J documentation, “Denoising Autoencoders”
  http://deeplearning.net/tutorial/dA.html

Administrivia

- Read over project updates—they all look great! (no notes)
  - Update #2 due on April 10 (1 week from today)
- **Final Presentations**
  - April 22, 23, and 24
  - **Part of your grade is your attendance—so please come even if you aren't presenting!**
  - **20 minutes to speak** (+2 for Q&A after)
  - 3-4 slots on April 22 and 24; 2 slots on April 23
  - Sign-up is **first-come, first-serve**: send me a DM with your 1st and 2nd choice and I'll try to slot you in