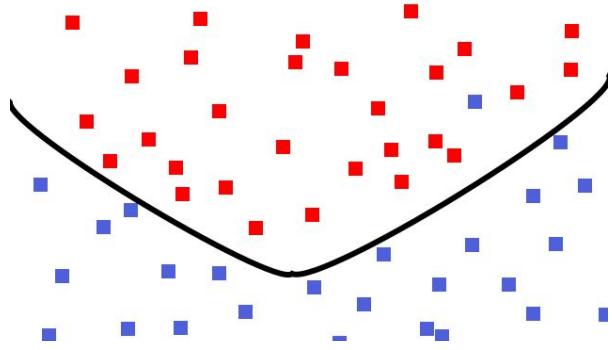


UNSUPERVISED LEARNING

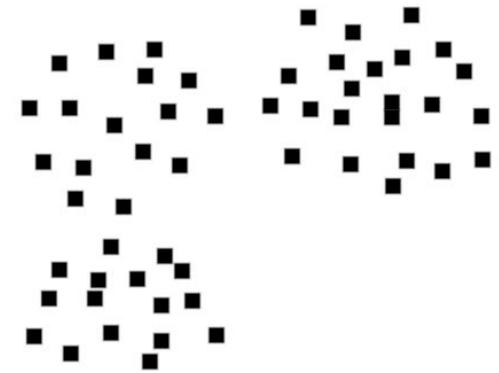
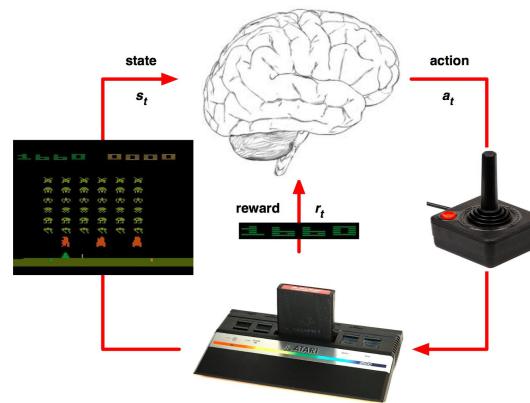
3 Modes of Learning



Supervised Learning



Reinforcement Learning

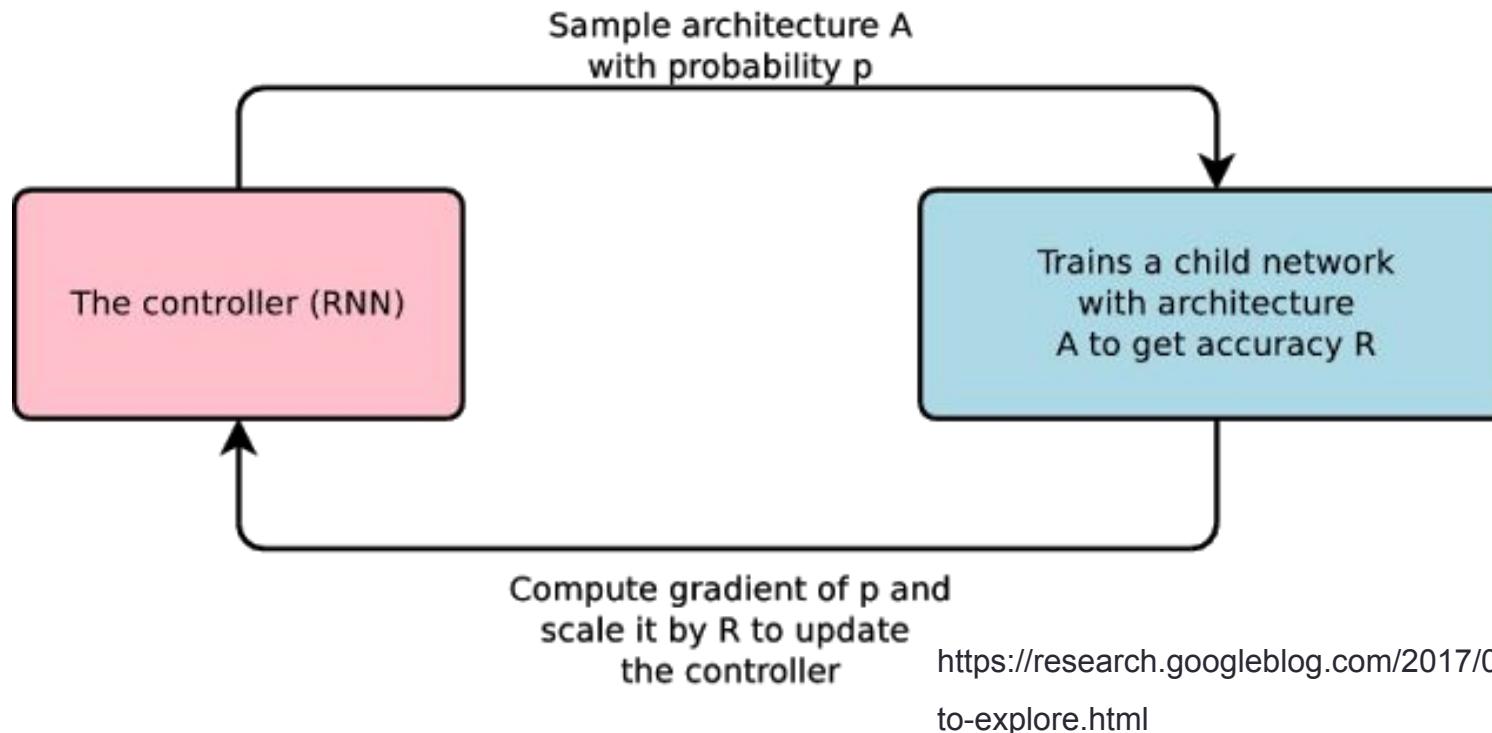


Unsupervised Learning



Model selection for deep architectures

- Tuning a network takes time
- Let machine learning learns how to tune a network
- Matches or outperforms ML experts performance

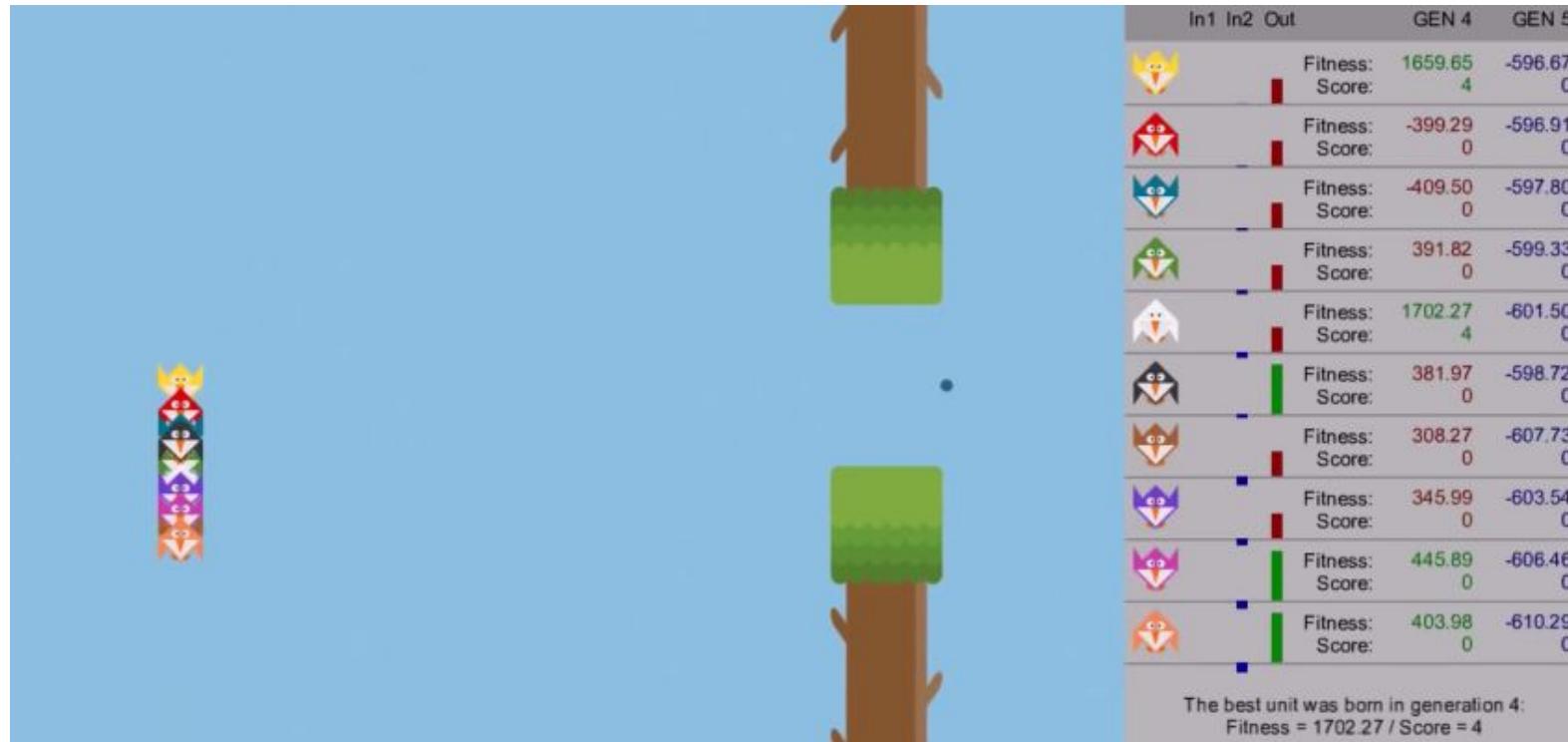


Genetic Algorithm for Reinforcement learning

- Updating based on the gradients for RL is hard
 - Credit assignment problem
 - Target network as a workaround for lowering variance
- Estimate the direction to minimize the loss by random perturbations
 - Perturb the weights and let the new agents roam in the environment
- Highly parallelizable

Flappy bird genetic algorithm

- <https://www.youtube.com/watch?v=aeWmdojEJf0>



Yann LeCun's comment



Yann LeCun

March 14, 2016 ·

Follow

Statement from a Slashdot post about the AlphaGo victory: "We know now that we don't need any big new breakthroughs to get to true AI"

That is completely, utterly, ridiculously wrong.

As I've said in previous statements: most of human and animal learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don't know how to make the cake.

We need to solve the unsupervised learning problem before we can even think of getting to true AI. And that's just an obstacle we know about. What about all the ones we don't know about?

#deeplearning #AI #AlphaGo

How Much Information Does the Machine Need to Predict?

Y LeCun

■ "Pure" Reinforcement Learning (cherry)

- ▶ The machine predicts a scalar reward given once in a while.
- ▶ **A few bits for some samples**

■ Supervised Learning (icing)

- ▶ The machine predicts a category or a few numbers for each input
- ▶ Predicting human-supplied data
- ▶ **10→10,000 bits per sample**

■ Unsupervised/Predictive Learning (cake)

- ▶ The machine predicts any part of its input for any observed part.
- ▶ Predicts future frames in videos
- ▶ **Millions of bits per sample**



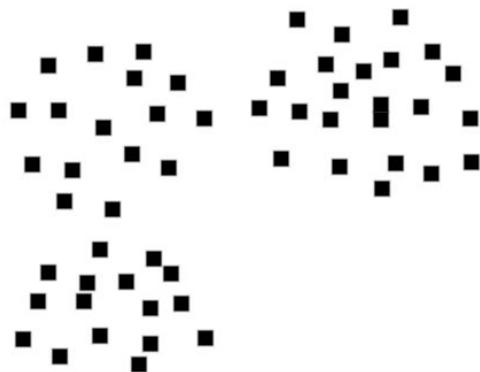
■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Unsupervised Learning

- Clustering
 - Introduction to probabilistic graphical models
 - LDA
- Autoencoders
 - Denoising Autoencoder
 - VAE
 - Evidence lower bound
 - Perceptual loss
 - Image prior
- Generative Adversarial Networks
- Random Cool stuff

Clustering

- Discover underlying structure of data



- Learn the hidden class labels in the data.

Simple clustering

- K-means
 - Find the means, and assign
- GMM
 - Find the means, covariance, then soft assignment
- K-means vs GMM
 - Spherical covariance vs any covariance
 - Hard vs soft assignment

Can we give priors knowledge to clustering?

Latent Dirichlet Allocation (LDA)

- Another method that learns the hidden labels.
- Mostly used for topic modeling and document classification
- A kind of probabilistic graphical model
- Let's cover some basics

Topic modeling motivation

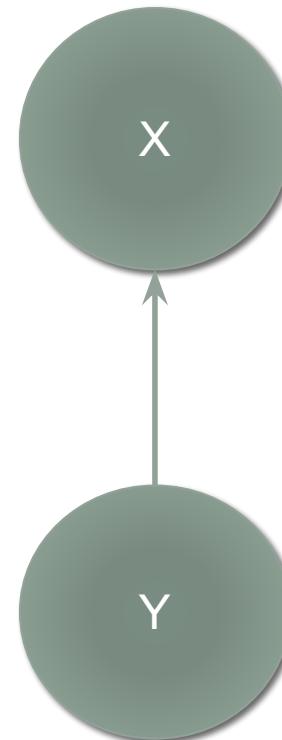
- I want to be able to give probabilities to words/sentences/documents.
- $P(\text{"Roses are red. Violets are blue."}) = ?$
- $P(X_1 = \text{Rosses}, X_2 = \text{are}, X_3 = \text{red}, \dots)$
 - This probability distribution is intractable

Introduction to graphical models

- Given rain in the morning predict sunny or cloudy in the afternoon

Graphical representation of the relationship

$P(X Y)$	$Y =$ Rain	$Y =$ No rain
$X =$ cloudy		
$X =$ sunny		

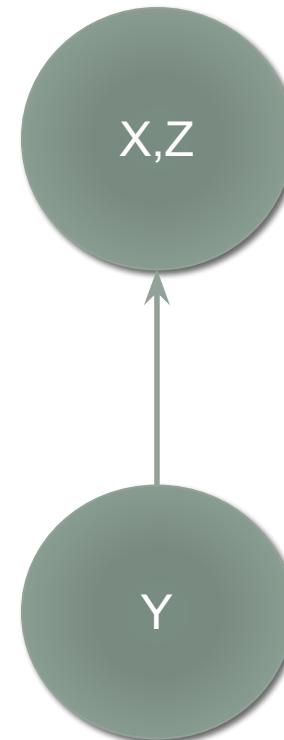


Curse of dimensionality

- Given rain in the morning predict sunny or cloudy and windy or cloudy in the afternoon

Graphical representation of the relationship

$P(X,Z Y)$	$Y =$ Rain	$Y =$ No rain
X = cloudy Z = windy		
X = sunny Z = windy		
X = cloudy Z = no wind		
X = sunny Z = no wind		

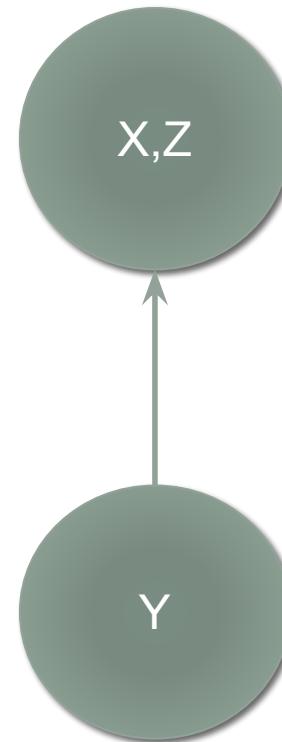


Curse of dimensionality

- If there are more variables, this table keeps getting bigger and bigger

Graphical representation of the relationship

$P(X,Z Y)$	$Y =$ Rain	$Y =$ No rain
X = cloudy Z = windy		
X = sunny Z = windy		
X = cloudy Z = no wind		
X = sunny Z = no wind		



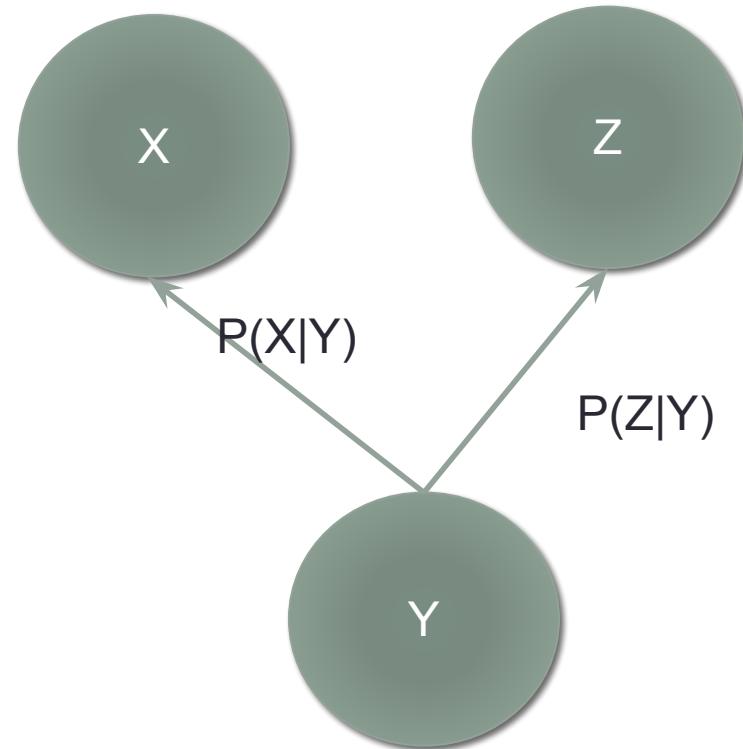
Naïve Bayes

- Assume conditional independence

$P(X Y)$	$Y =$ Rain	$Y =$ No rain
$X =$ cloudy		
$X =$ sunny		

$P(Z Y)$	$Y =$ Rain	$Y =$ No rain
$Z =$ windy		
$Z =$ no wind		

Graphical representation of the relationship

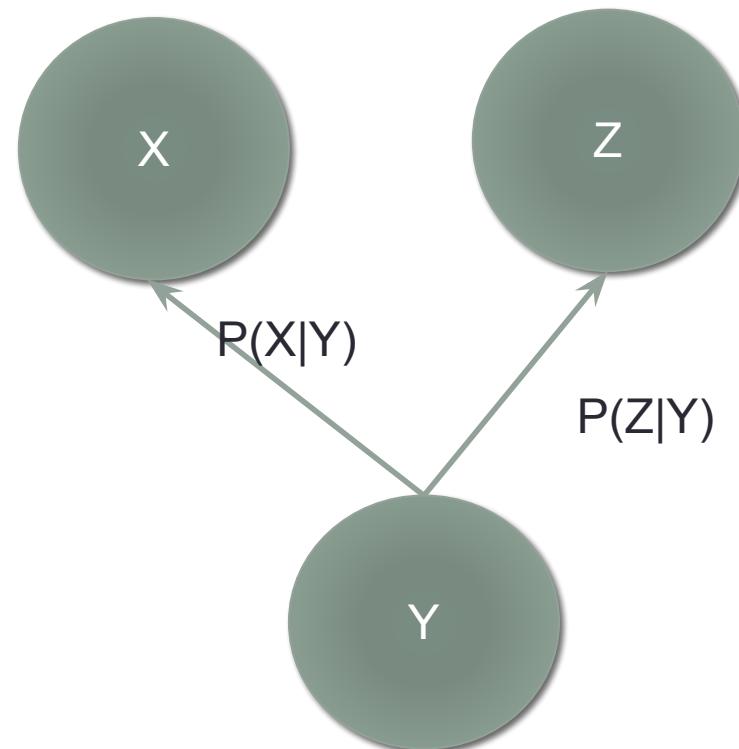


Probabilistic models and independence assumptions

- Most probabilistic models will assume some independence assumption in order to reduce the parameter space

- Graphical representations helps summarize the relationship
 - Arrows denote a conditional relationship

Graphical representation of the relationship



Probabilistic models and independence assumptions

- Also implies
- $P(X,Y,Z) = P(Y)P(X|Y)P(Z|Y)$

Graphical representation of the relationship

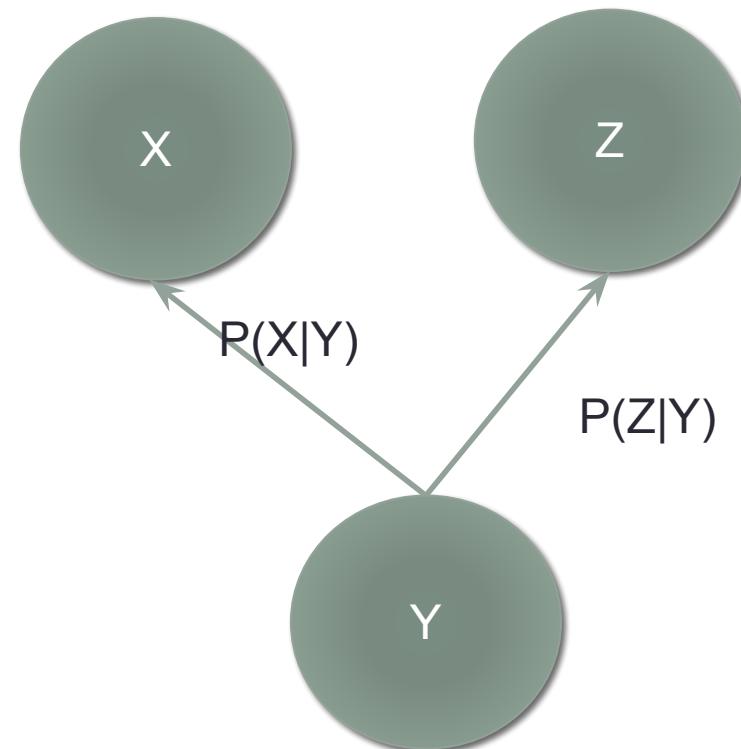


Plate notation

- Sometimes you have too many relationships.

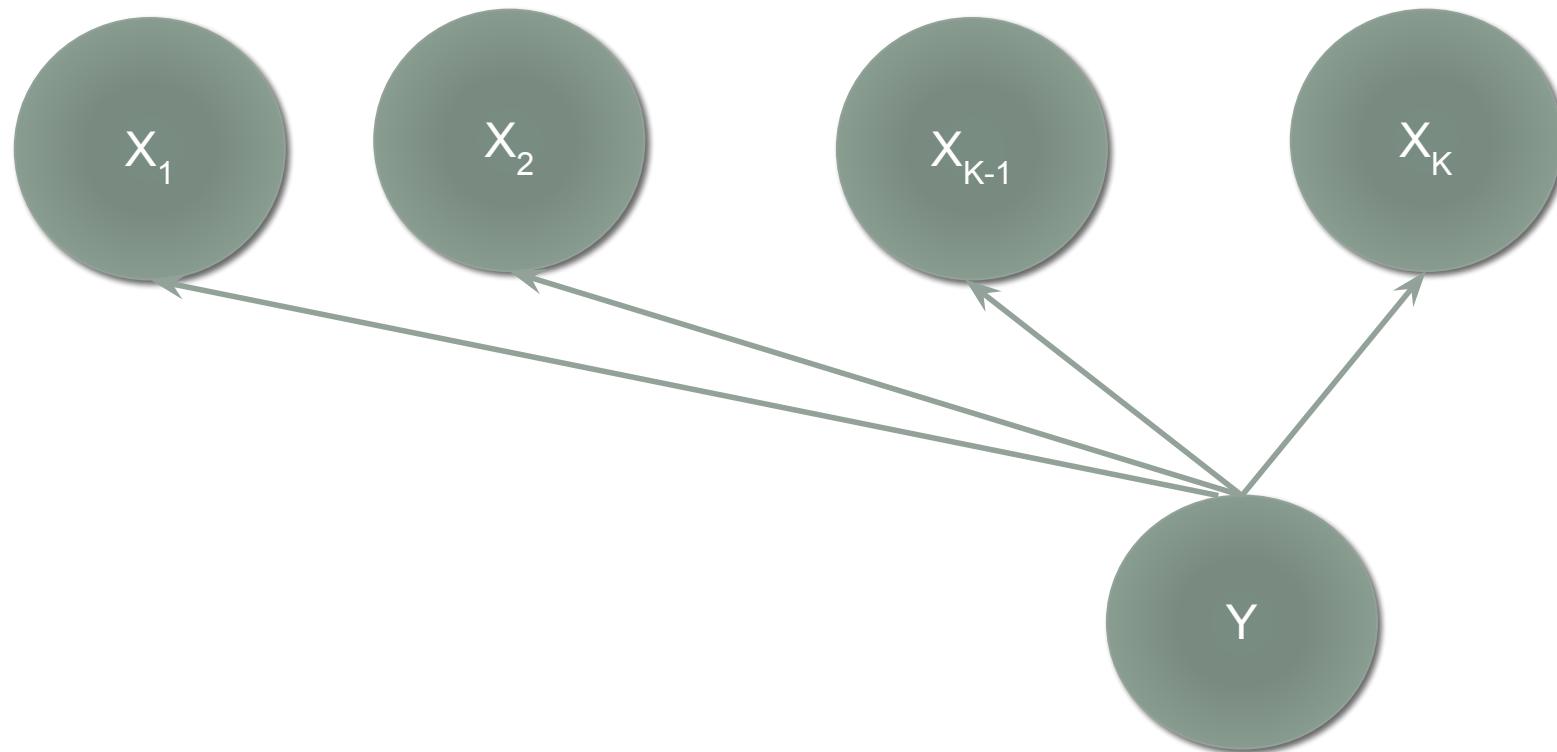


Plate notation

- Summarize by using a square box with number

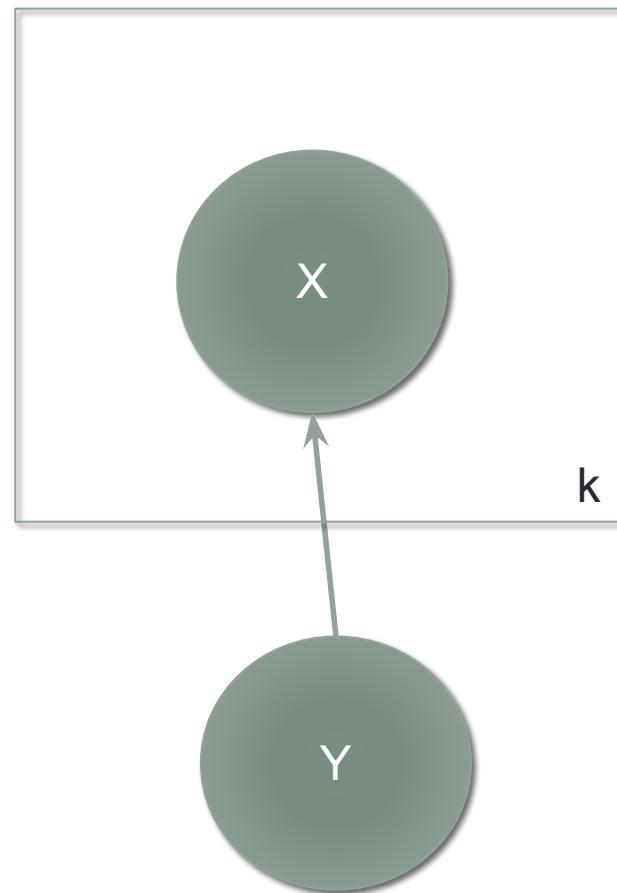
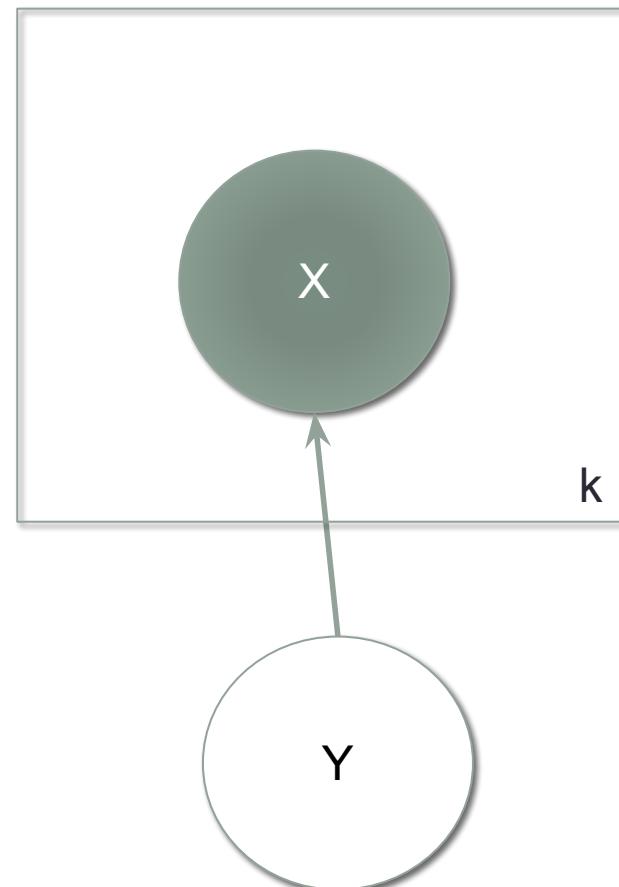


Plate notation

- Summarize by using a square box with number
- Dark circles are observed variables
- White circles are **latent variables**

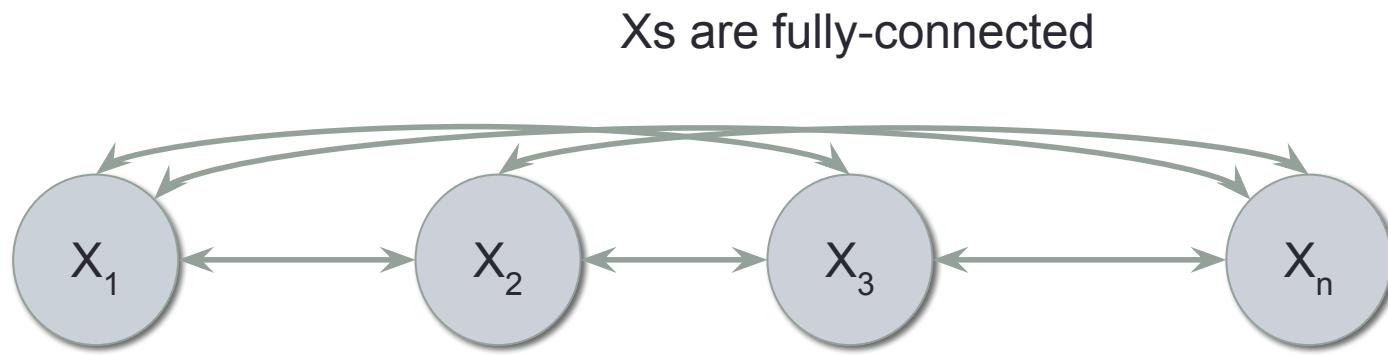


Topic modeling motivation

- I want to be able to give probabilities to words/sentences/documents.
- $P(X_1 = \text{Rosses}, X_2 = \text{are}, X_3 = \text{red}, \dots)$
 - This probability distribution is intractable
- Assume independence
 - $P(X_1) P(X_2) P(X_3) \dots$
 - Lost all relational structure. Can we do better?
 - Assume words come from topics. How?

Graphical view of language modeling

- $P(\text{"Roses are red. Violets are blue."}) = ?$

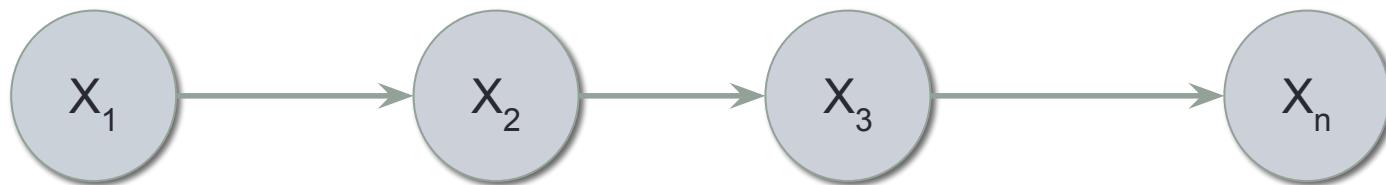


$$P(X_1 = \text{Roses}, X_2 = \text{are}, \dots, X_n = \text{blue})$$

Graphical view of language modeling

- $P(\text{"Roses are red. Violets are blue."}) = ?$

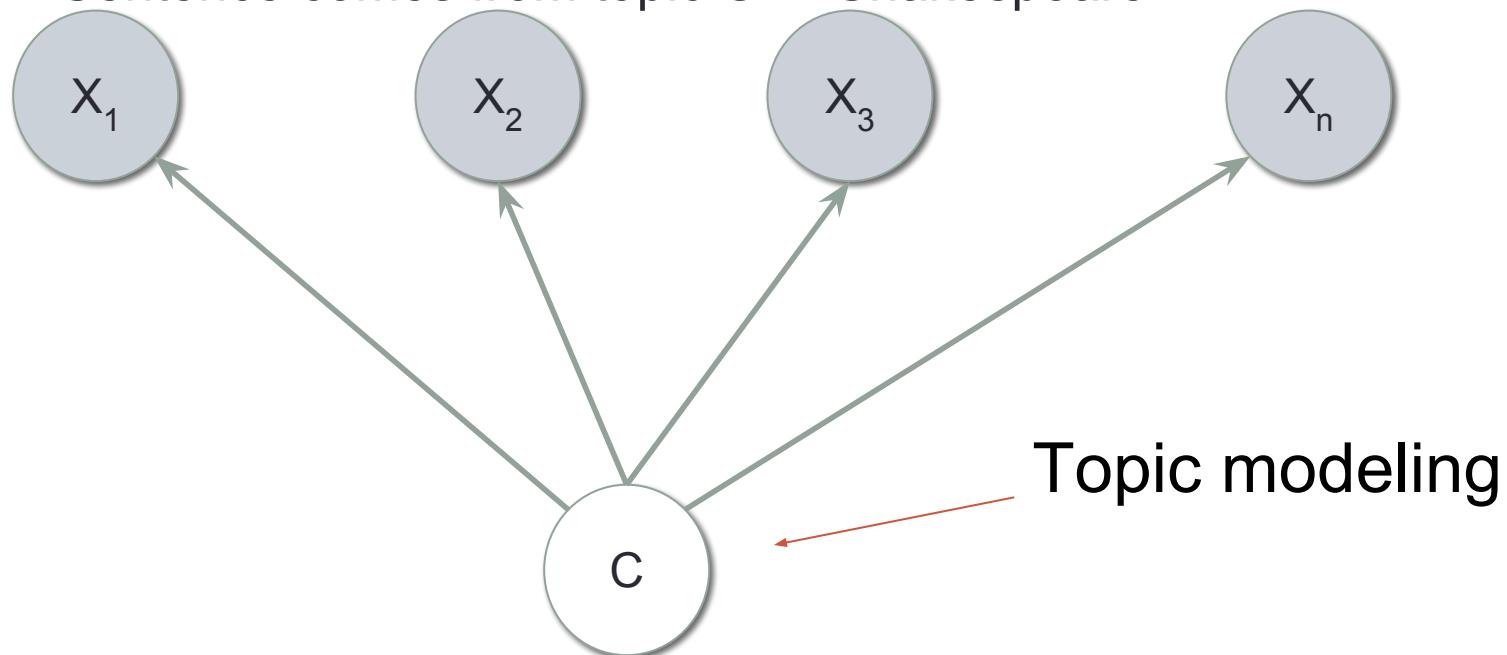
With markov assumption



$$P(X_1 = \text{Roses})P(X_2 = \text{are} | X_1 = \text{Roses}) P(X_3 = \text{red}) \dots P(X_n = \text{blue} | X_{n-1} = \text{are})$$

Graphical view of language modeling

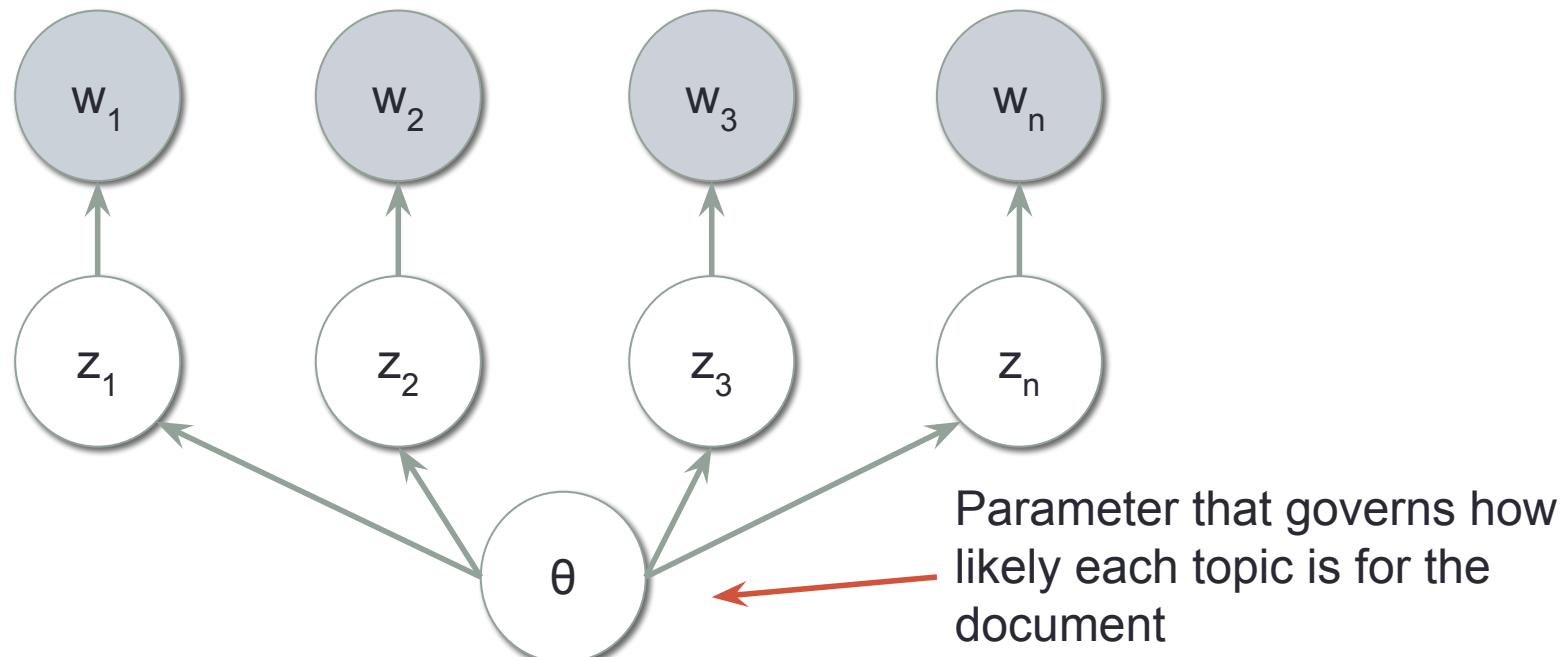
- $P(\text{"Roses are red. Violets are blue."}) = ?$
- Let's assume words are generated from a topic
 - Sentence comes from topic C = Shakespeare



$$P(X_1 = \text{Roses} | c = \text{Shakespeare}) P(X_2 = \text{are} | c = \text{Shakespeare}) \dots \\ P(X_n = \text{blue} | c = \text{Shakespeare}) P(c = \text{Shakespeare})$$

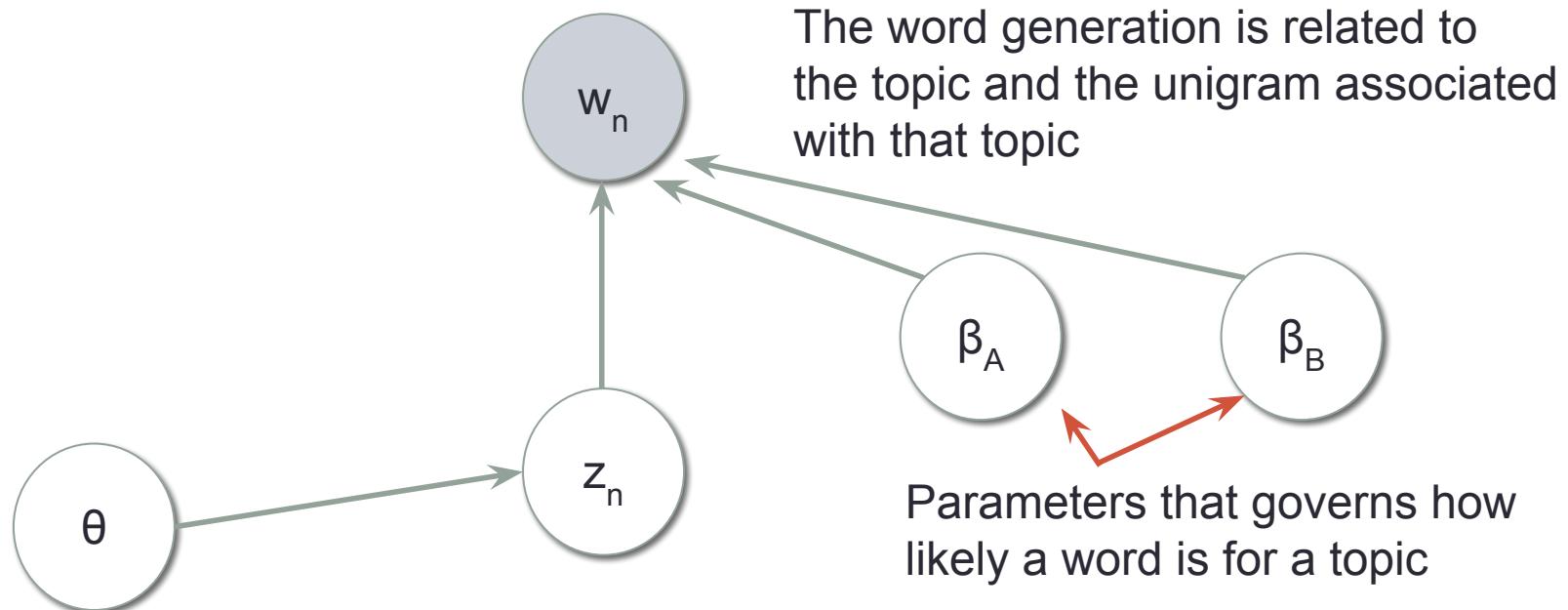
Multiple topics

- Most documents have multiple topics. Our model assumes 1 document 1 topic.
 - Let a document be a mixture of topics. Each word has its own topic, z .
 - $P(w) = P(z = A) P(w | z = A) + P(z = B) P(w | z = B)$
 - $P(z = A) + P(z = B) = 1, \quad \theta = P(z = A)$



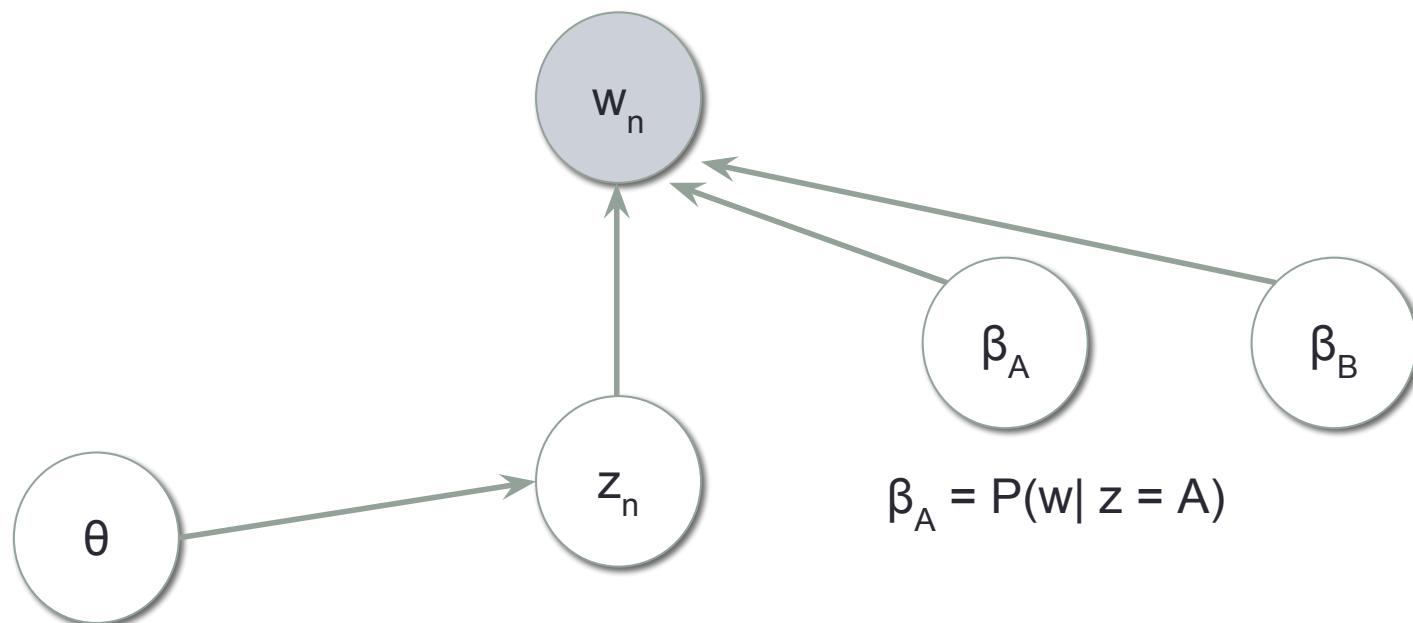
Topic modeling

- Most documents have multiple topics. Our model assumes 1 document 1 topic.
 - Let a document be a mixture of topics (language model interpolation). Each word has its own topic, z .
 - $P(w) = P(z = A) P(w | z = A) + P(z = B) P(w | z = B)$
 - $P(z = A) + P(z = B) = 1, \quad \theta = P(z = A) \quad \beta_A = P(w | z = A)$



Graphical model and generation

- You can generate a sample from a graphical model by following the arrows



Graphical model and generation

- Given θ, β_A, β_B



A = 0.3
B = 0.7

Cat = 0.5
Dog = 0.5

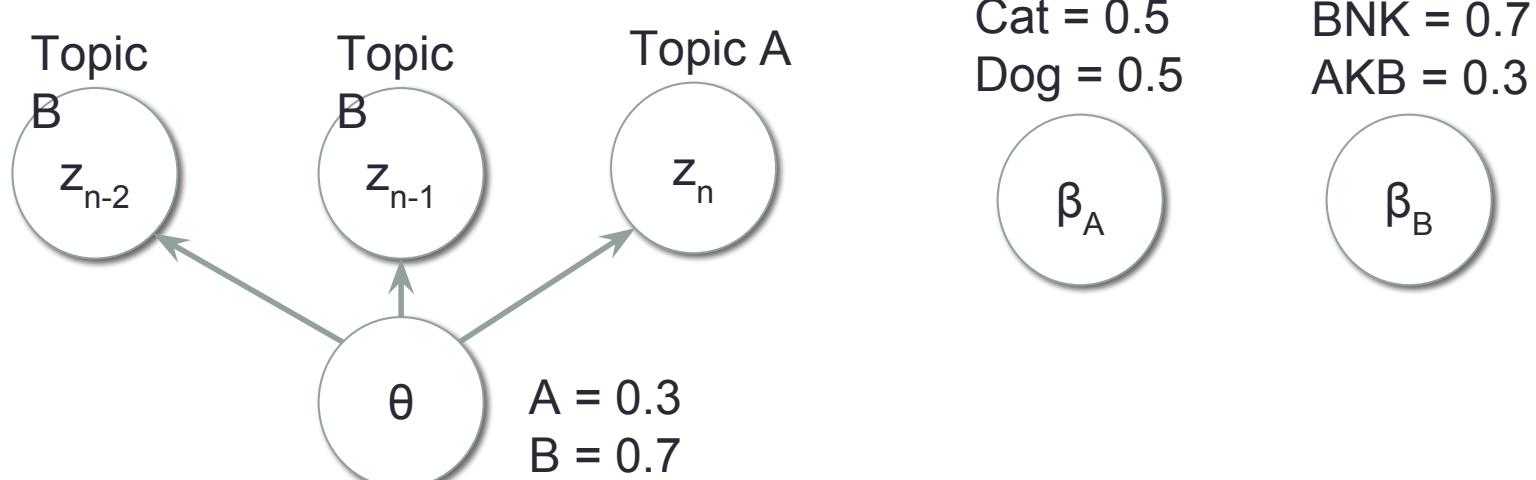


BNK = 0.7
AKB = 0.3



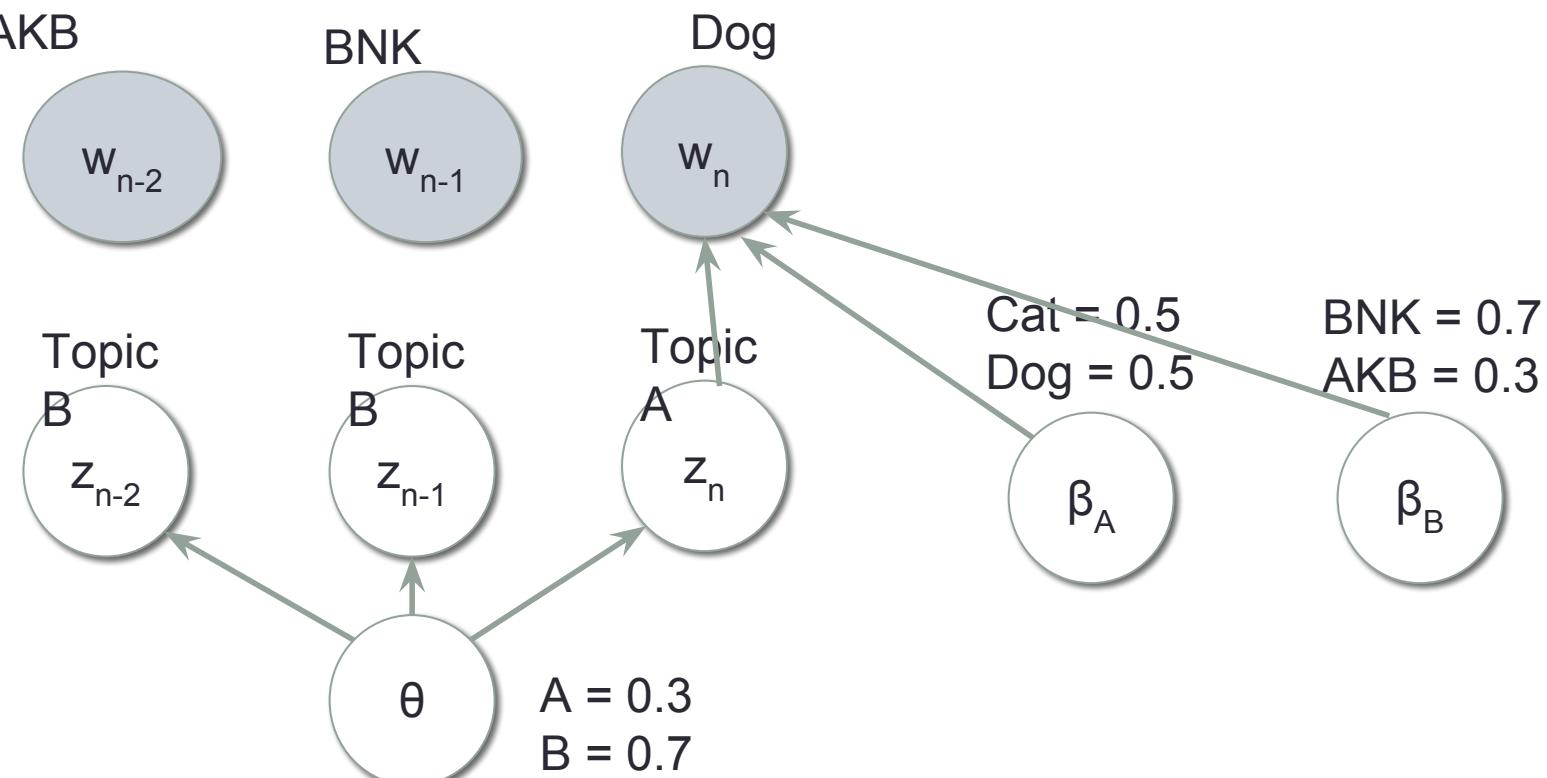
Graphical model and generation

- Given θ, β_A, β_B
- Generate (randomly create) z from θ



Graphical model and generation

- Given θ, β_A, β_B
- Generate (randomly create) z from θ
- Generate w_n from z_n, β_A, β_B



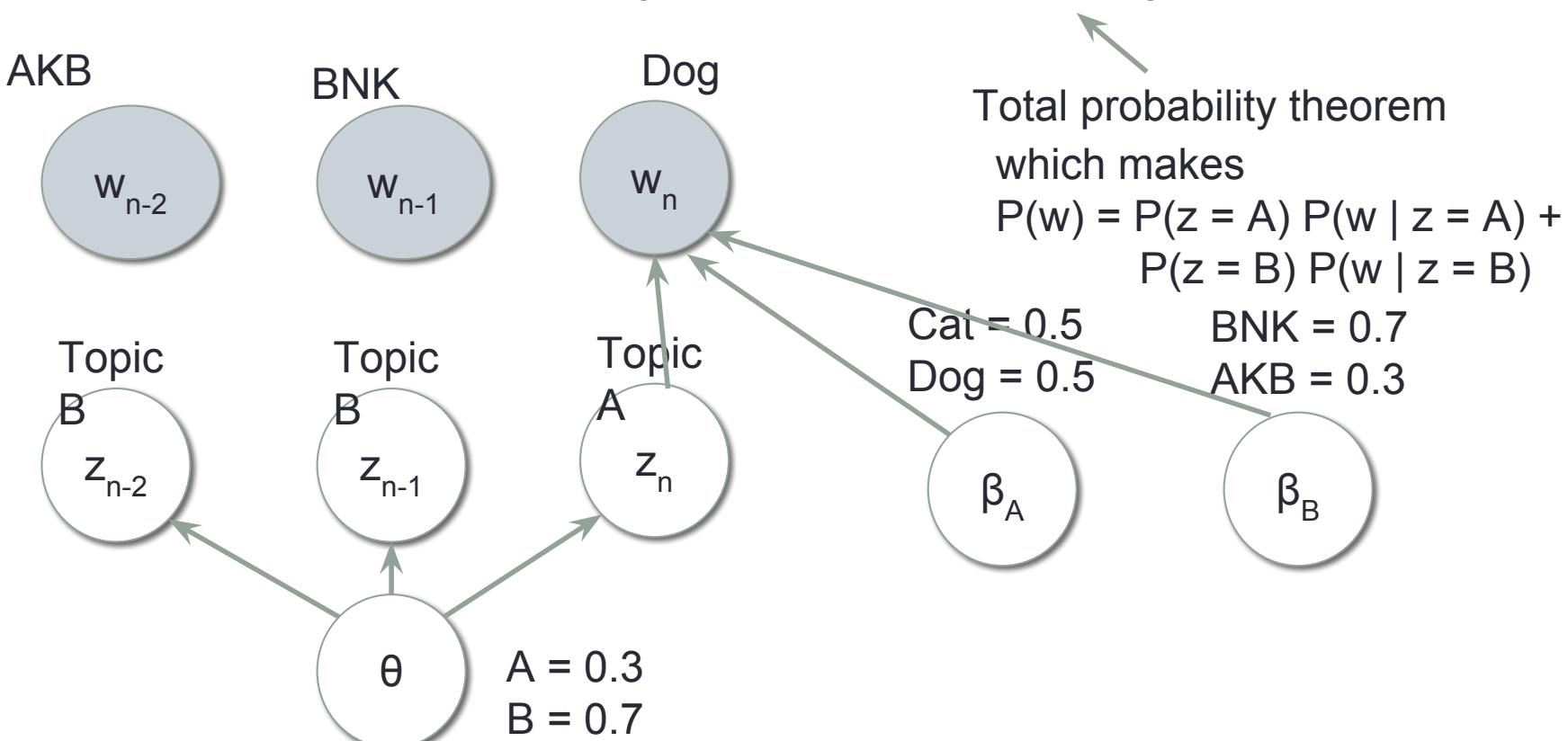
Graphical model and generation

How likely a sentence is likely to be generated follows this generation process

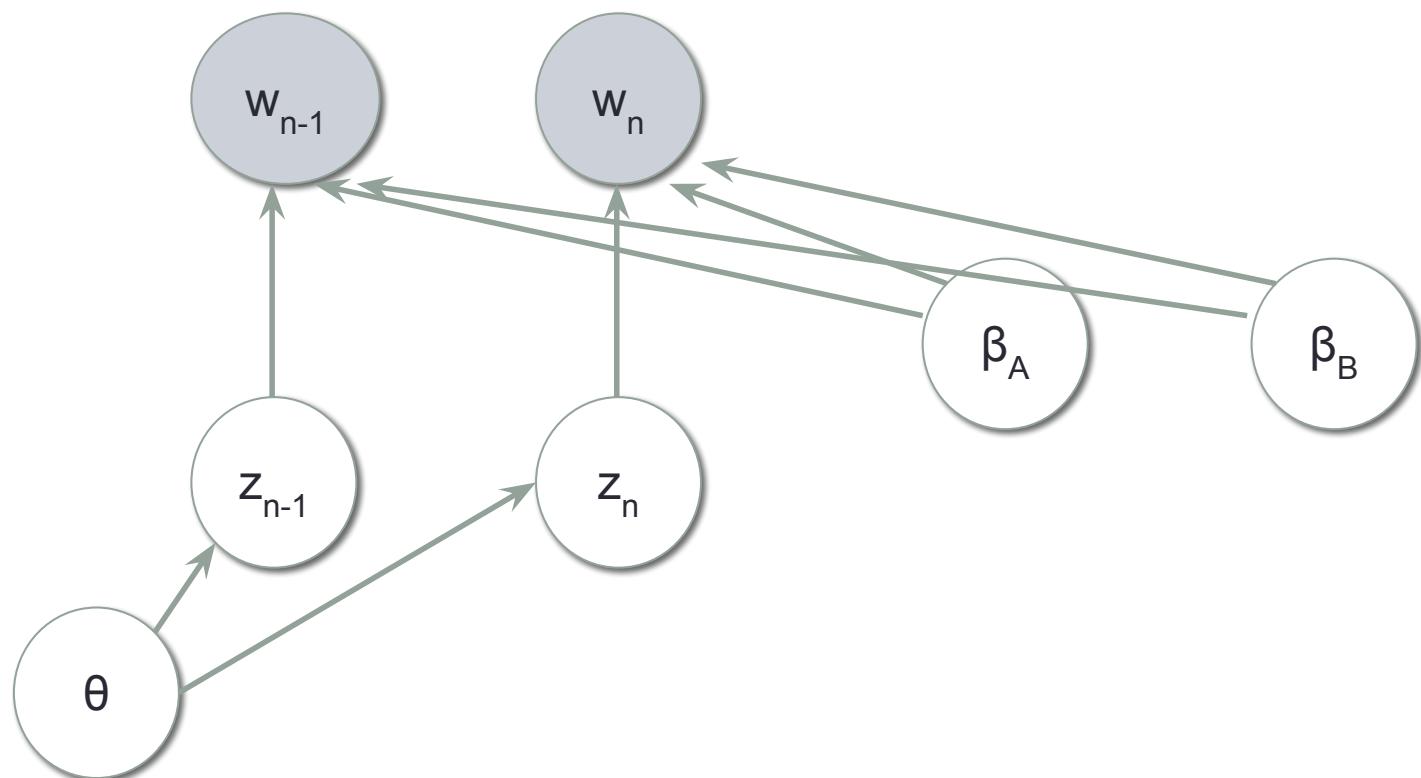
$$P(AKB, BNK, Dog, B, B, A) = P(B)P(B)P(A)P(AKB|B)P(BNK|B)P(Dog|A)$$

Note

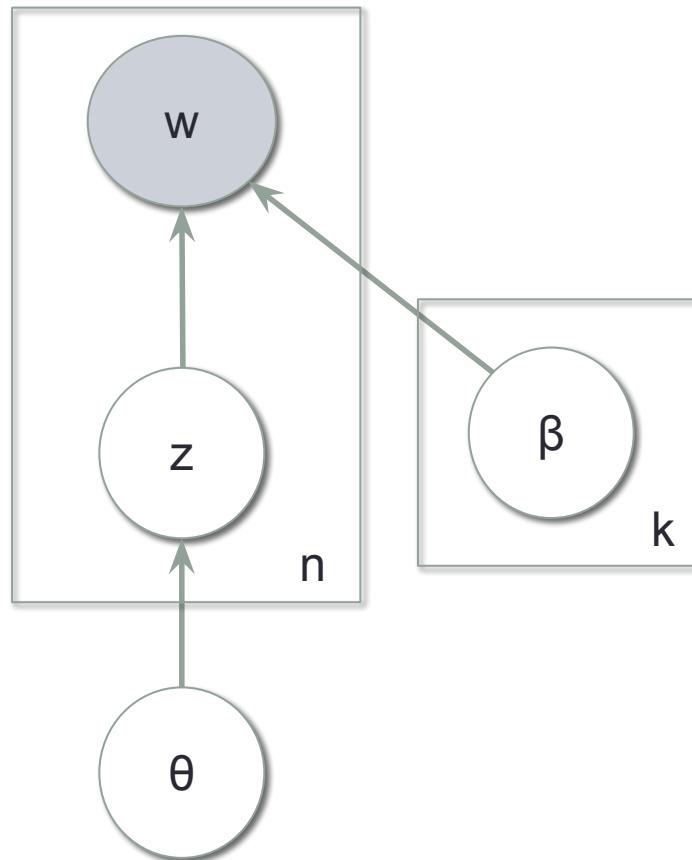
$$\begin{aligned} P(AKB, BNK, Dog) &= P(AKB, BNK, Dog, A, A, A) + P(AKB, BNK, Dog, A, A, B) \\ &\quad P(AKB, BNK, Dog, A, B, A) + P(AKB, BNK, Dog, A, B, B) + \dots \end{aligned}$$



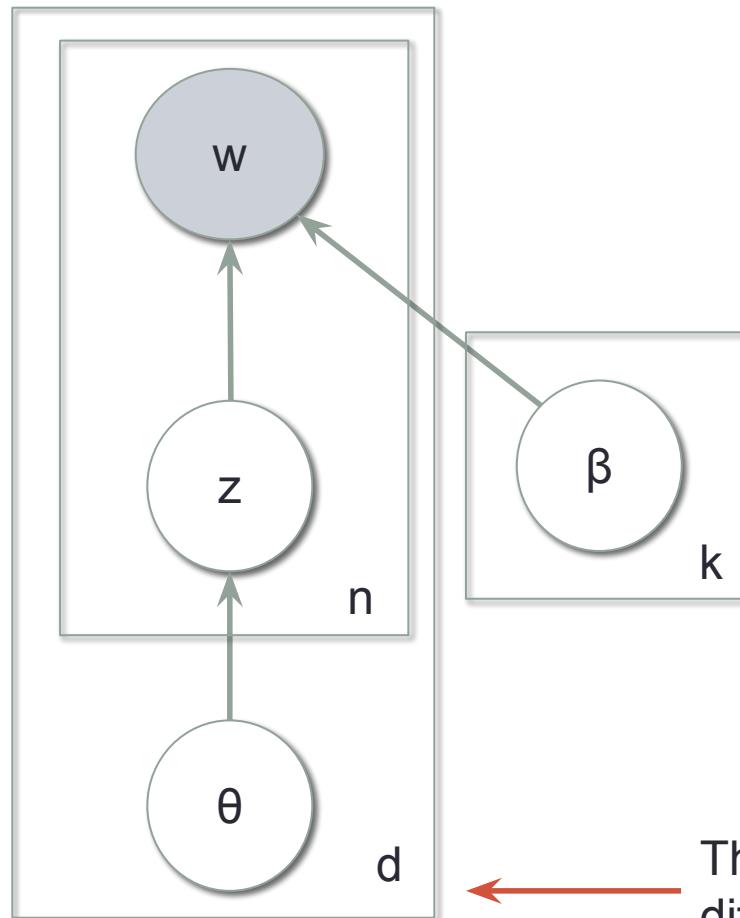
Topic modeling with plate notation



Topic modeling with plate notation

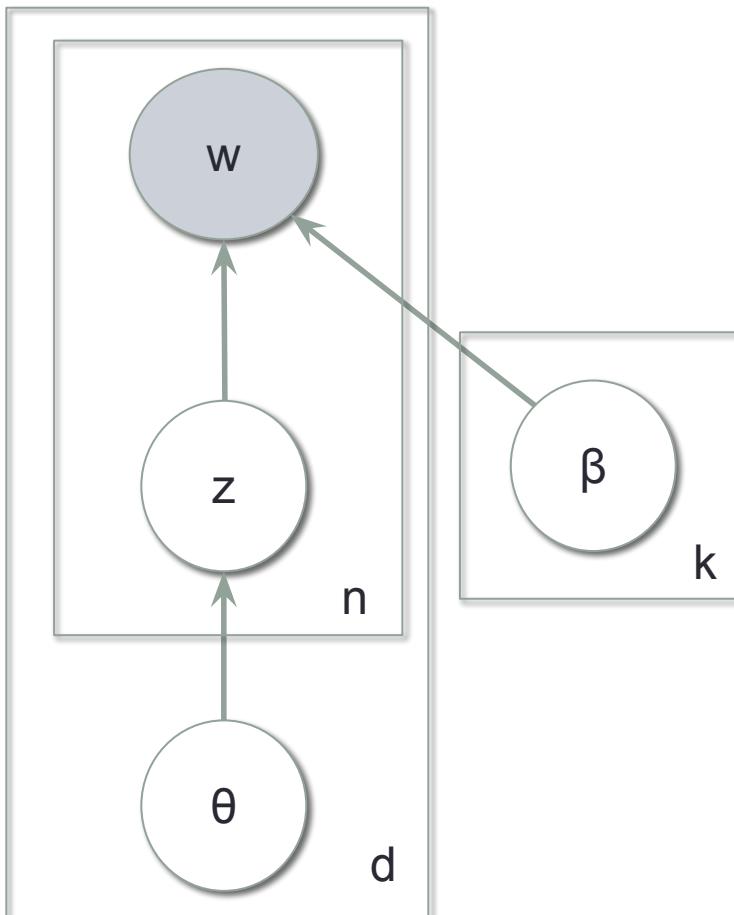


Topic modeling with plate notation



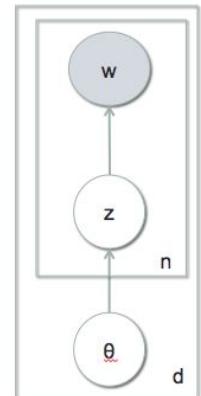
There are d documents each with
different topic distributions

Topic modeling with plate notation



Called pLSA
probabilistic Latent Semantic Analysis

Note: if you look at other textbooks, you will see a slightly different picture



Learning topic latent model parameters

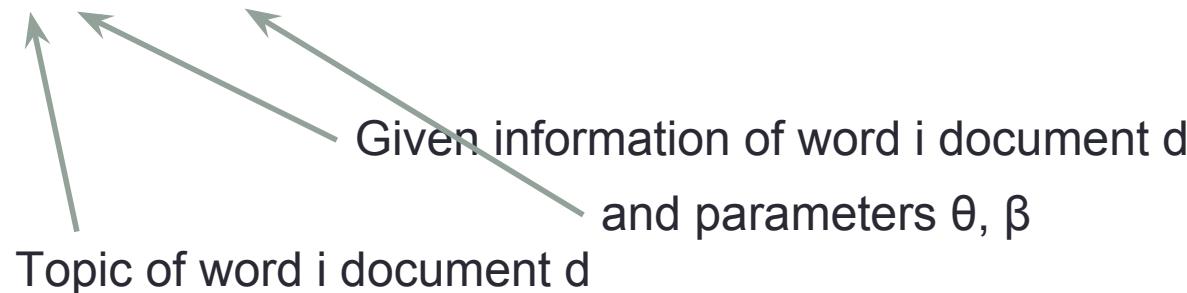
- How to find θ and β ?
 - “Cat,Dog,BNK”, “Cat, cat, cat, BNK”, “Dog, dog, BNK, dog, AKB”
- If we know, the latent topic z for each word we can use the counts
 - “Cat_A, Dog_A, BNK_B”
- $P(\text{Cat}|A) = \frac{\text{count}(\text{Cat}_A)}{\text{count}(A)}$
- $P_1(A) = \frac{\text{count}(A)}{\text{count}(\text{all words in document 1})}$
- But we don't know the topic z for each word

Expectation maximization (EM)

- A method to iteratively maximize the likelihood of a model on training data
 - Expectation step (E step) – guess latent variables from model parameters (get soft counts)
 - Maximization step (M step) – re-estimate model parameters from latent variables (counts)

E-step

- Find an estimate for the latent variable given parameters θ, β
- $p(z_{di} | w_{di}, \theta, \beta)$



E-step

- Find an estimate for the latent variable given parameters θ, β

$$p(z_{di}|w_{di}, \theta, \beta) = \frac{p(z_{di}, w_{di}, \theta, \beta)}{\sum_{z'=1}^k p(z'_{di}, w_{di}, \theta, \beta)}$$

$p(w, \theta, \beta)$

E-step

- Find an estimate for the latent variable given parameters θ, β

$$p(z_{di}|w_{di}, \theta, \beta) = \frac{p(z_{di}, w_{di}, \theta, \beta)}{\sum_{z'=1}^k p(z'_{di}, w_{di}, \theta, \beta)}$$
$$= \frac{\theta_z|d\beta_w|z}{\sum_{z'=1}^k \theta_{z'}|d\beta_w|z'}$$

Index di for z and w dropped for clarity

E-step

- Find an estimate for the latent variable given parameters θ, β

$$p(z_{di}|w_{di}, \theta, \beta) = \frac{p(z_{di}, w_{di}, \theta, \beta)}{\sum_{z'=1}^k p(z'_{di}, w_{di}, \theta, \beta)}$$
$$= \frac{\theta_{z|d}\beta_{w|z}}{\sum_{z'=1}^k \theta_{z'|d}\beta_{w|z'}}$$


P(Word is from topic A | word is cat from document 1)
Probability that the word is from each topic. Use as counts

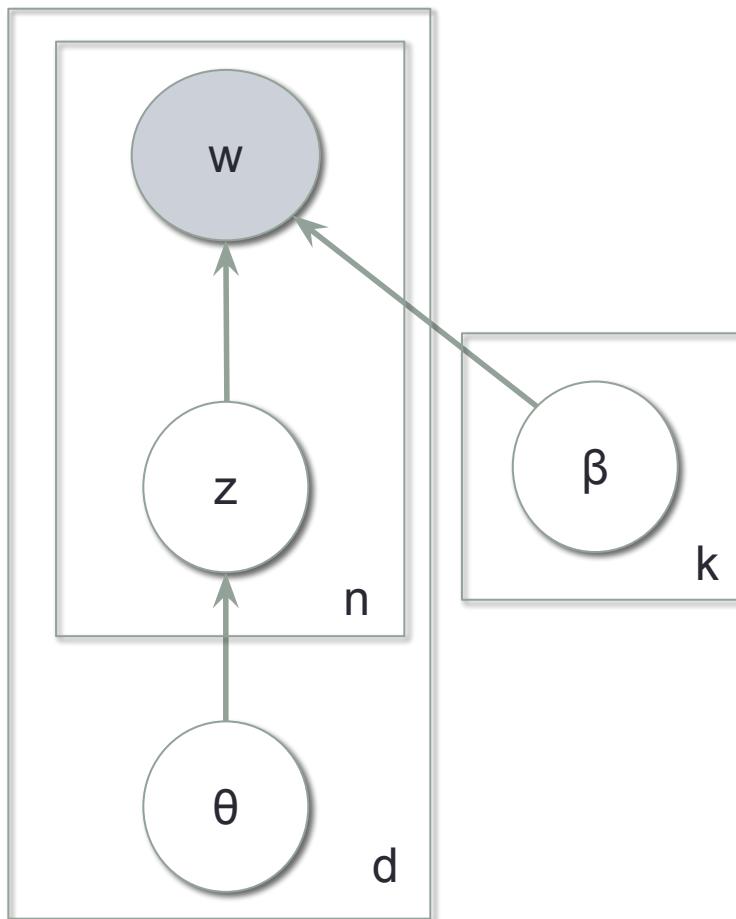
M-Step

- Instead of real counts use $P(z_{di})$ as the topic label
 - $P(\text{Cat}|A) = \frac{\text{count}(\text{Cat}_A)}{\text{count}(A)}$
 - $P_1(A) = \frac{\text{count}(A)}{\text{count}(\text{all words in document } 1)}$

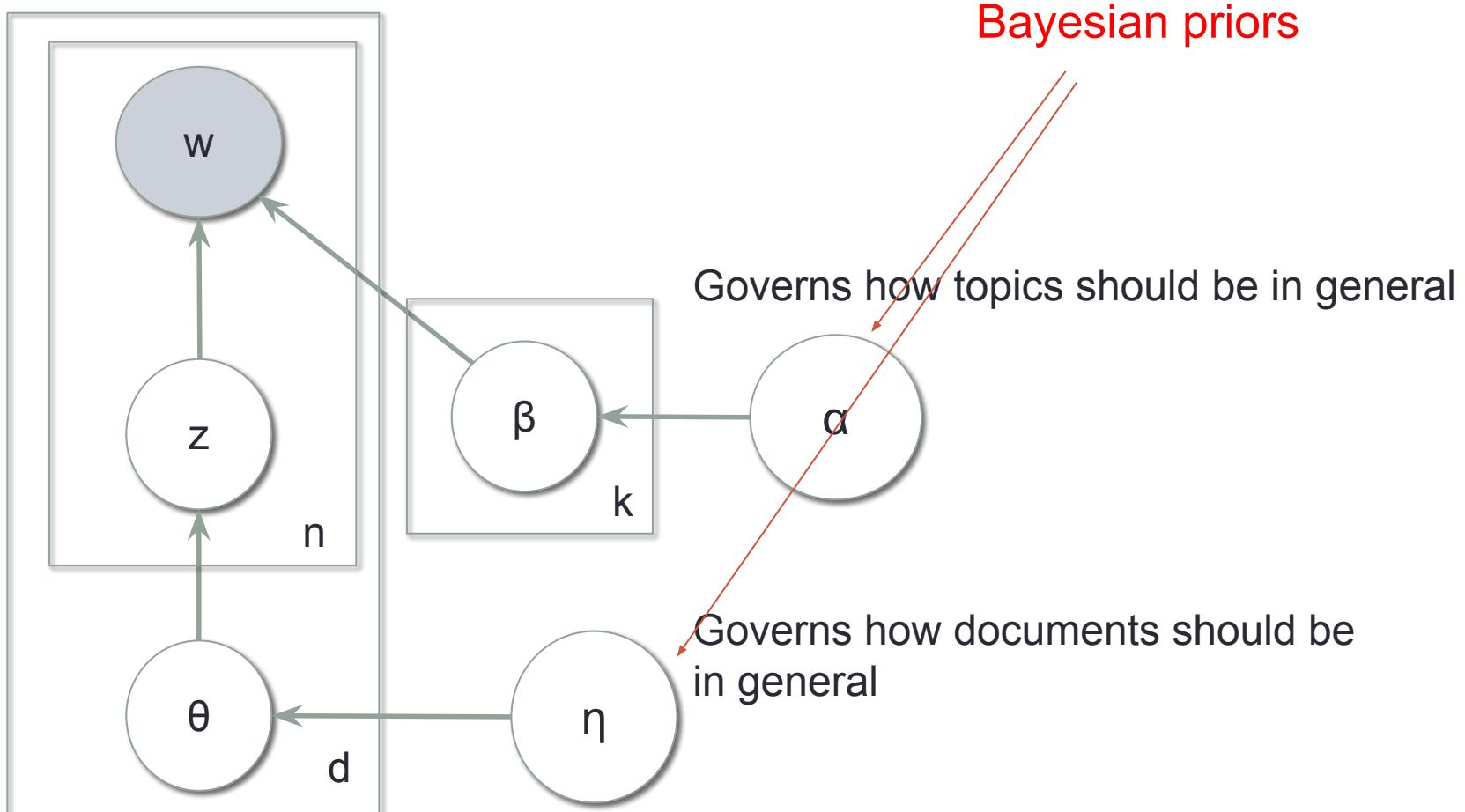
pLSA

- pLSA automatically clusters words into topic unigrams
 - Requires user to specify number of topics
- Automatically learn document representation based on the learned topics
 - $\text{DocA} = [0.7 \ 0.3]$ $\text{DocB} = [0.2 \ 0.8]$ $\text{DocC} = [0.5 \ 0.5]$
- Overfits easily to data outside of the training set
 - Nothing that ties all document together
 - A document from a document collection should be have topic distributions that are similar
- Solution: LDA (Latent Dirichlet Allocation)

pLSA



LDA



α and η

- α is a **Dirichlet distribution**
- Or a distribution of distributions...
- Example: rolling a die
- $P(x=1) = 0.3, P(x=2) = 0.1, \dots$
- This is a distribution. (**Multinomial distribution**)
- But what if I have a bag of dices, each with different distributions.
- α tells me the probability of what kind of die I will get

Dirichlet distribution

- pdf

Parameters giving preference to each side of die/topic

$$f(x_1, \dots, x_K; \alpha_1, \dots, \alpha_K) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^K x_i^{\alpha_i - 1}$$

Probability of each side of die, probability of each topic

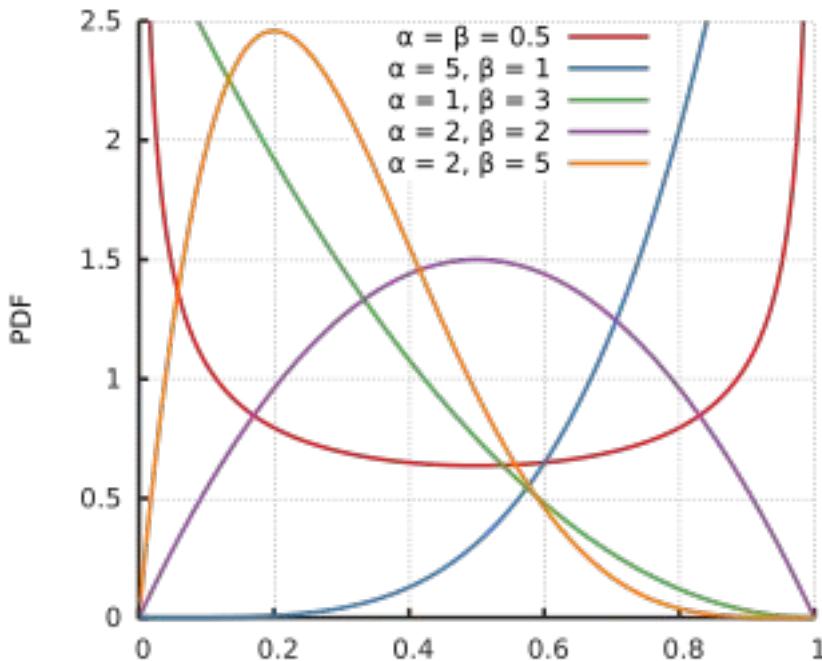
Dirichlet distribution

- pdf

Parameters giving preference to each side of die/topic

$$f(x_1, \dots, x_K; \alpha_1, \dots, \alpha_K) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^K x_i^{\alpha_i - 1}$$

Probability of each side of die, probability of each topic



$$p(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

Beta distribution
A prior for binomial trials

Parameter learning

- How do we actually learn these parameters and latent variables?
 - Gibbs sampling
 - Variational methods

Topics

gene 0.04
dna 0.02
genetic 0.01
...

life 0.02
evolve 0.01
organism 0.01
...

brain 0.04
neuron 0.02
nerve 0.01
...

data 0.02
number 0.02
computer 0.01
...

Documents

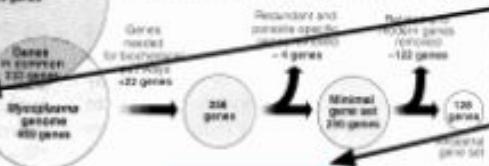
Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

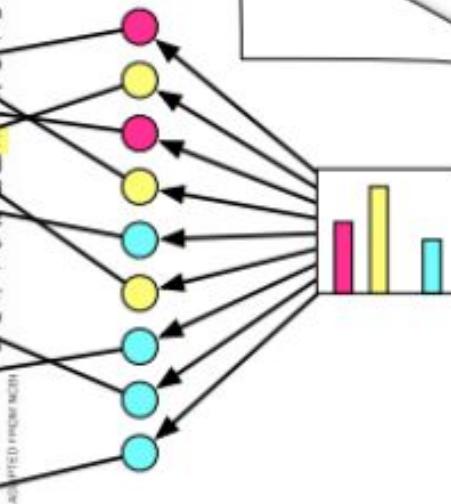
Although the numbers don't match precisely, those predictions

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Umeå University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a matter of numbers alone; particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing all



Topic proportions and assignments



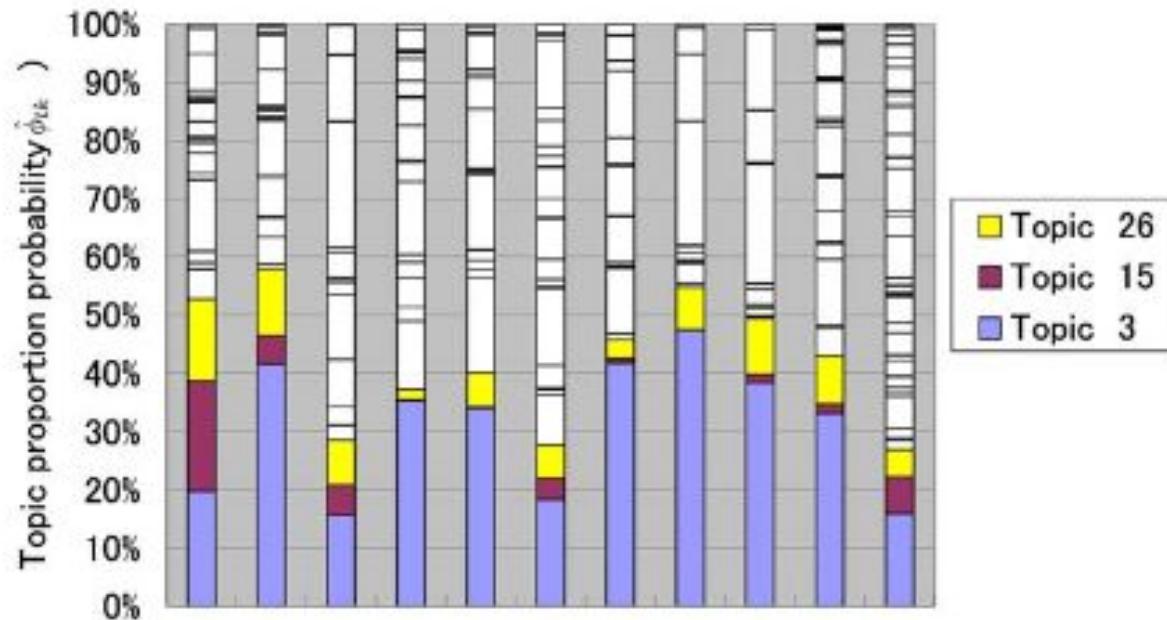
Example usage

- Lots of lectures. Can we discover the topics? Can we classify the lectures? No topic labels given

Top 10 high probability nouns in word probabilities of Topics 3 ($\theta_{k=3}$), 15 ($\theta_{k=15}$), and 26 ($\theta_{k=26}$).

Topic 3 (~ classical mechanics)	Topic 15 (~astronomy)	Topic 26 (~ (time) unit)
m	Light	Percent
Energy	Degrees	Time
Force	Angle	Dollars
Mass	Frequency	Times
Point	Energy	Minutes
Velocity	Direction	Day
Direction	Waves	Bit
v	Sun	Year
Times	Star	Hour
Speed	Speed	Half

Topic distribution of each document



Unsupervised topic modeling for real estate

Can we learn real estate characteristics from unstructured data?

คอนโดหุ้สไตร์ล้องกฤษ แห่งแรกในเข้า
ใหญ่ ที่ติด ถ.ธนารักษ์ มากที่สุด 1
ห้องนอน 1 ห้องน้ำ 1 ห้องนั่งเล่น
พร้อมห้องครัวแยกเป็นสัดส่วน

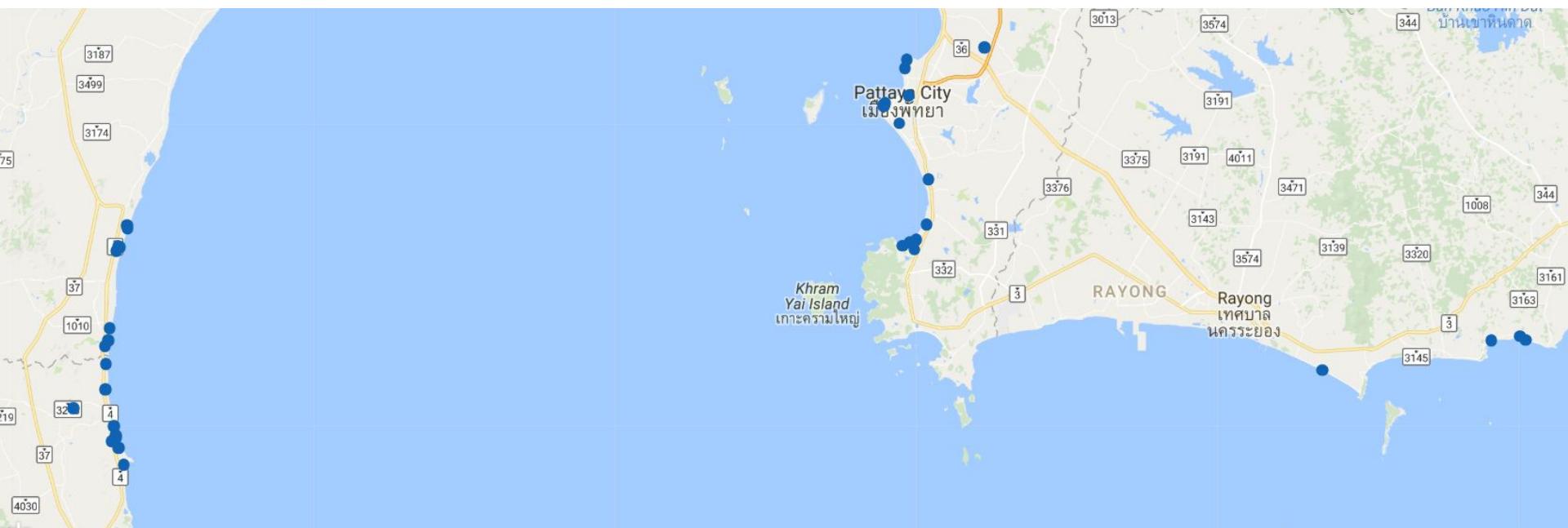
Just give it a bunch of descriptions



LDA Examples

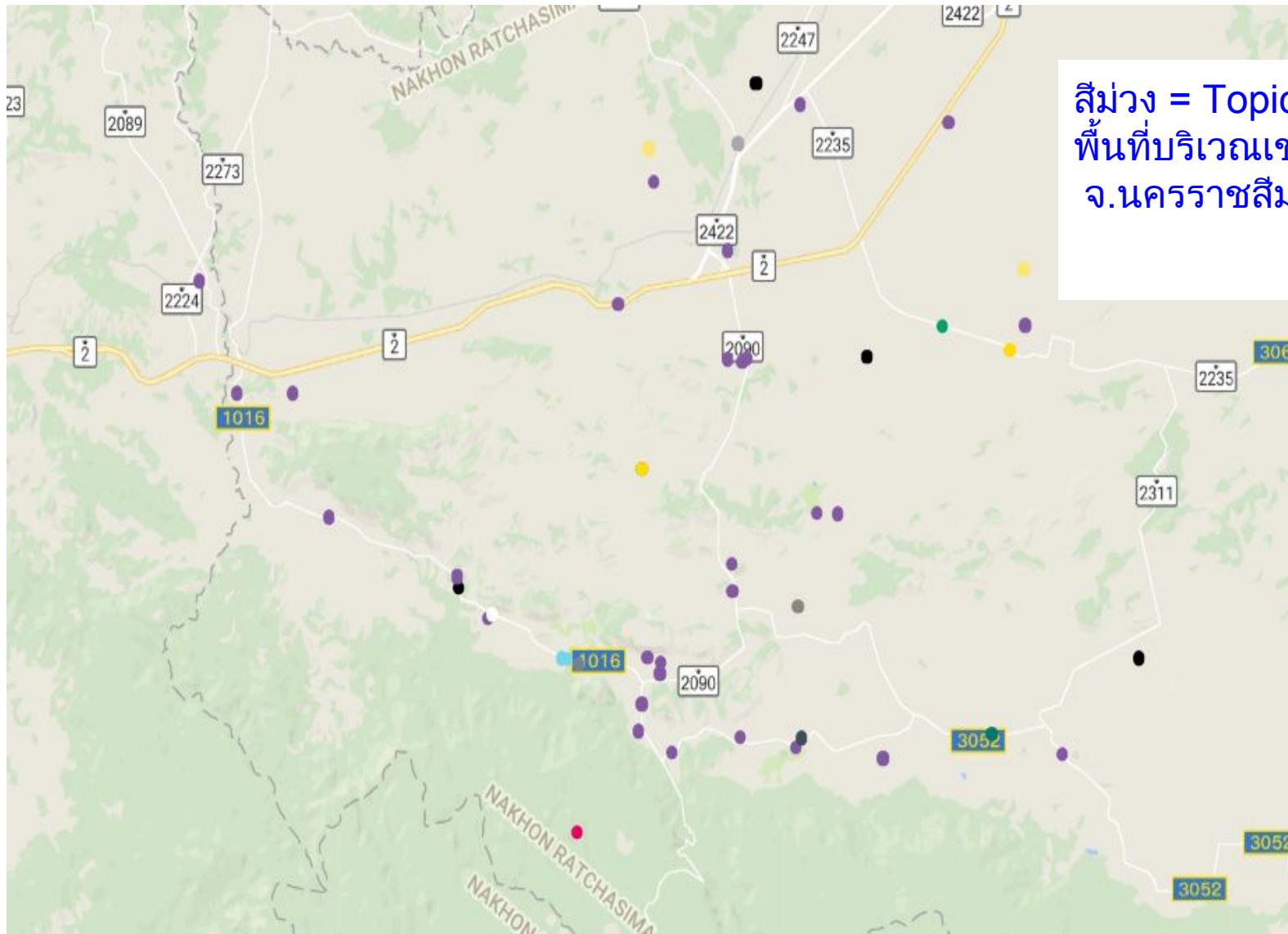
Topic 28

0.068*"วิว" + 0.058*"ทะเล" + 0.038*"คอนโด" + 0.029*"หัว" + 0.027*"คอนโดมิเนียม" + 0.025*"มองเห็น" + 0.023*"ทัศนียภาพ" + 0.022*"ชายหาด"



Topic 9

0.071*"ธรรมชาติ" + 0.031*"บรรยายกาศ" + 0.028*"ร่มรื่น" + 0.027*"บ้าน" + 0.025*"ท่ามกลาง" + 0.025*"สวน" + 0.025*"สัมผัส" + 0.021*"พื้นที่"



ลีม่วง = Topic 9
พื้นที่บริเวณเข้าใหญ่
จ.นครราชสีมา

Topic 40

0.115*"ระดับ" + 0.066*"เห็นอ" + 0.046*"หร" + 0.031*"ทำเล" + 0.026*"ชีวิต" + 0.026*"ใชชีวิต" + 0.016*"สไตล์" + 0.016*"สะท้อน"

Topic 17

0.077*"พื้นที่" + 0.060*"ออกแบบ" + 0.045*"โล่ง" + 0.039*"โปร่ง" + 0.038*"ใช้สอย" + 0.020*"ประโยชน์" + 0.018*"ห้อง" + 0.017*"อาคาร"

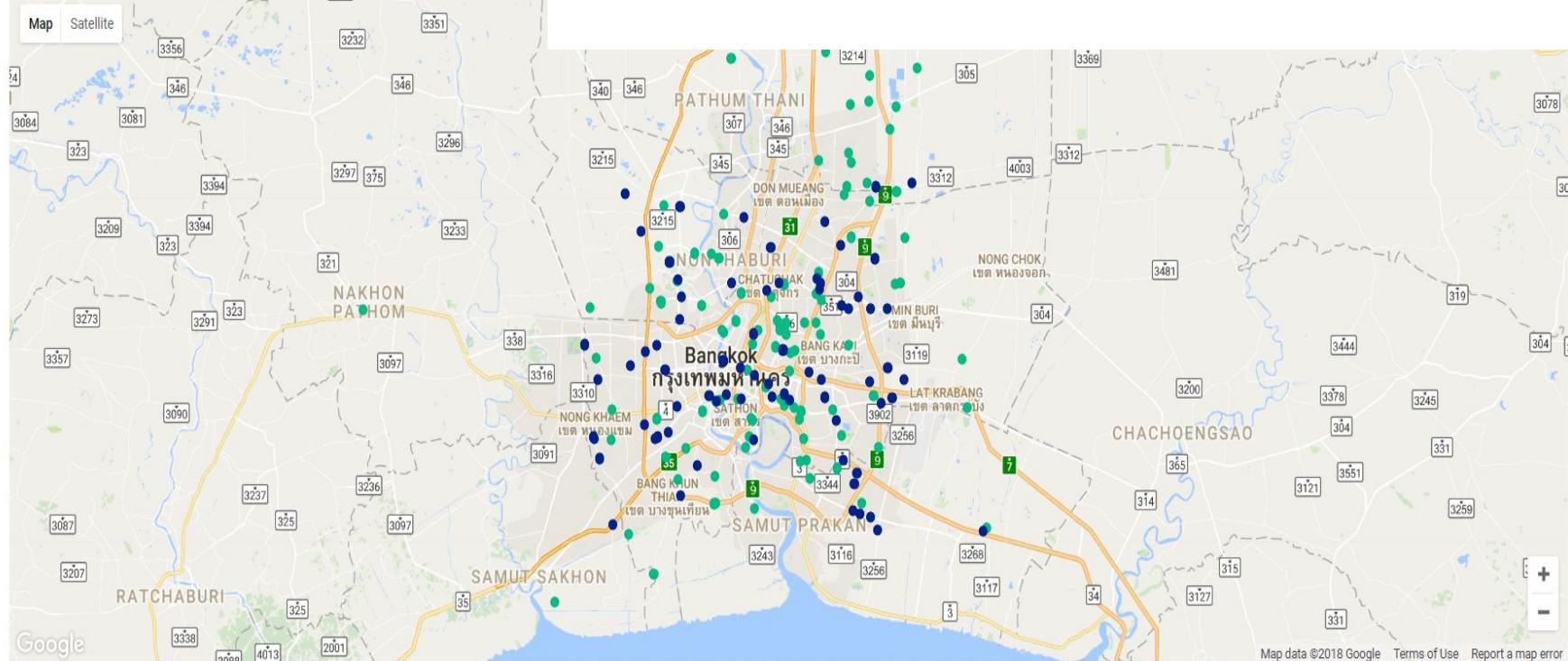


Select All Type Unselect All Type default

บ้านเดี่ยว บ้านแฝด ทาวน์เฮาส์ คอนโดมิเนียม อาคารพาณิชย์ โรงแรมพัฟฟิค ที่ดินเปล่า ทาวน์โฮม

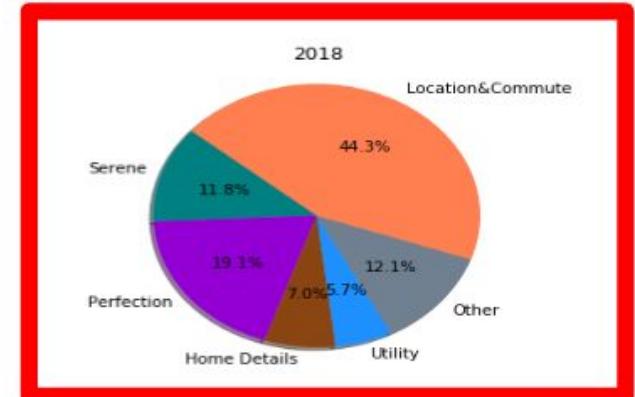
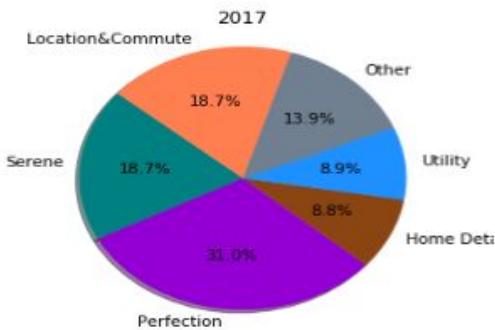
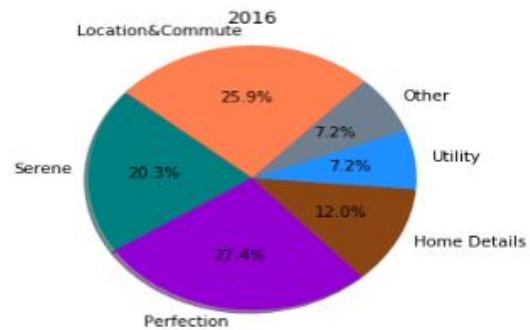
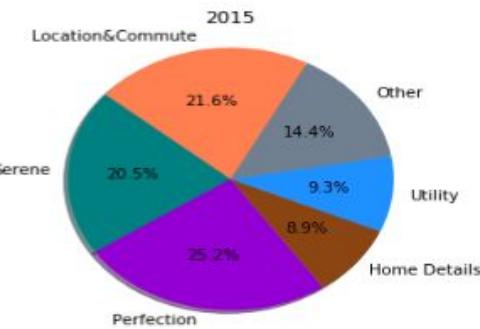
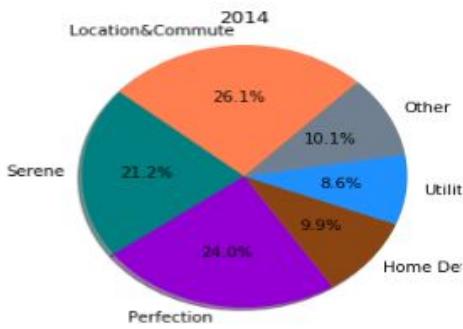
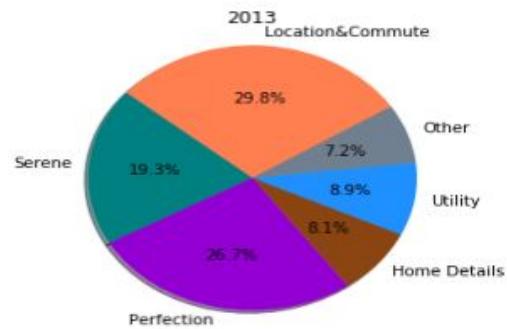
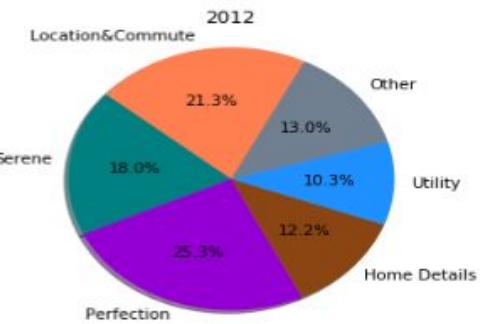
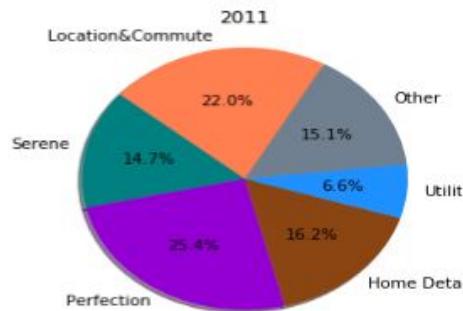
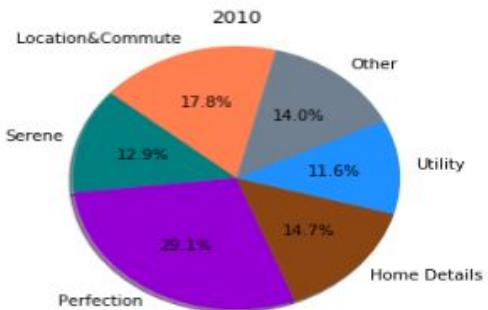
Select All Cluster Unselect All Cluster

cluster 0 cluster 1 cluster 2 cluster 3 cluster 4
 cluster 14 cluster 15 cluster 16 cluster 17 cluster 18
 cluster 27 cluster 28 cluster 29 cluster 30 cluster 31



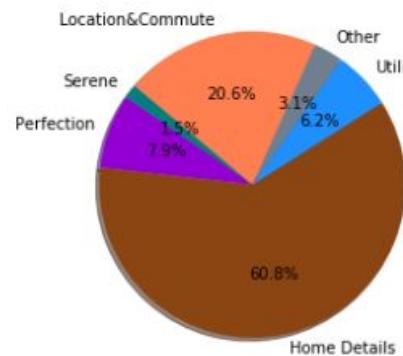
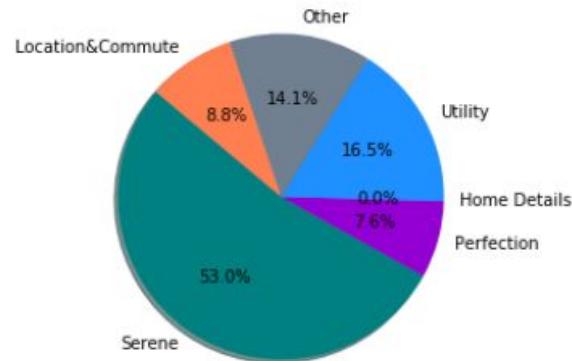
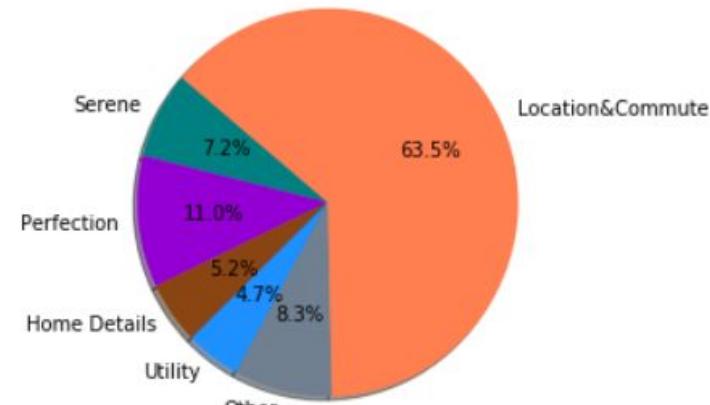
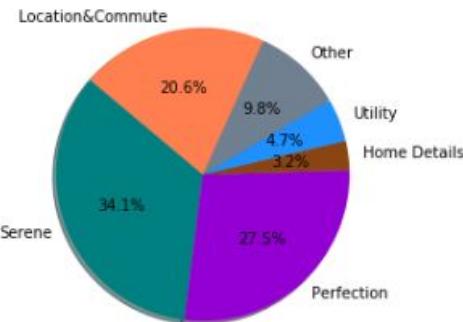
สิน้ำเงินเข้ม = โครงการที่มี Topic 40 อุ่นมาก (หร, ระดับ)
สีเขียว = โครงการที่มี Topic 17 (โครงการทั่วไป)

Time and advertisement trends



Advertisement niche of each developer

Each real estate developer has its own style



Discovering K

- In K-means, GMMs, LDA, we need K, the number of latent classes.
- Can we find it automatically?

Chinese Restaurant Process

- A random process is analogous to seating customers in a restaurant with infinite tables.
 - Each table has infinite seats
- First person sits in the first table
- n^{th} person sits at a table based on
 - Join existing table according to
 - number of person in that table, and table characteristics
 - Join a new empty table αc (some hyperparameter)



Chinese Restaurant Process

- After an assignment to a table, the seat is fixed. The table distribution is also updated.
 - CRP automatically starts a new class when it thinks there should be a new class.
 - Hyperparameter c controls how often to start a new class.
-
- Note: This is theoretically nice, but for most applications just pick a K .

Unsupervised Learning

- Clustering
 - Introduction to probabilistic graphical models
 - LDA
- Autoencoders
 - Denoising Autoencoder
 - VAE
 - Evidence lower bound
- Generative Adversarial Networks
- Random Cool stuff

“Unsupervised” Learning

- Given an image -> Tell you right away it's a cat without anyone labeling anything?

NO

- What does it do exactly?

Can we somehow utilize unlabeled data to help with some task we care about?

First Unsupervised Learning Idea

- Suppose we have a magic function f that maps an image to two numbers



First Unsupervised Learning Idea

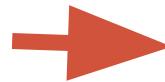
- Suppose we have a magic function f that maps an image to two numbers



(4.1, 5.0)



(8.1, 8.0)



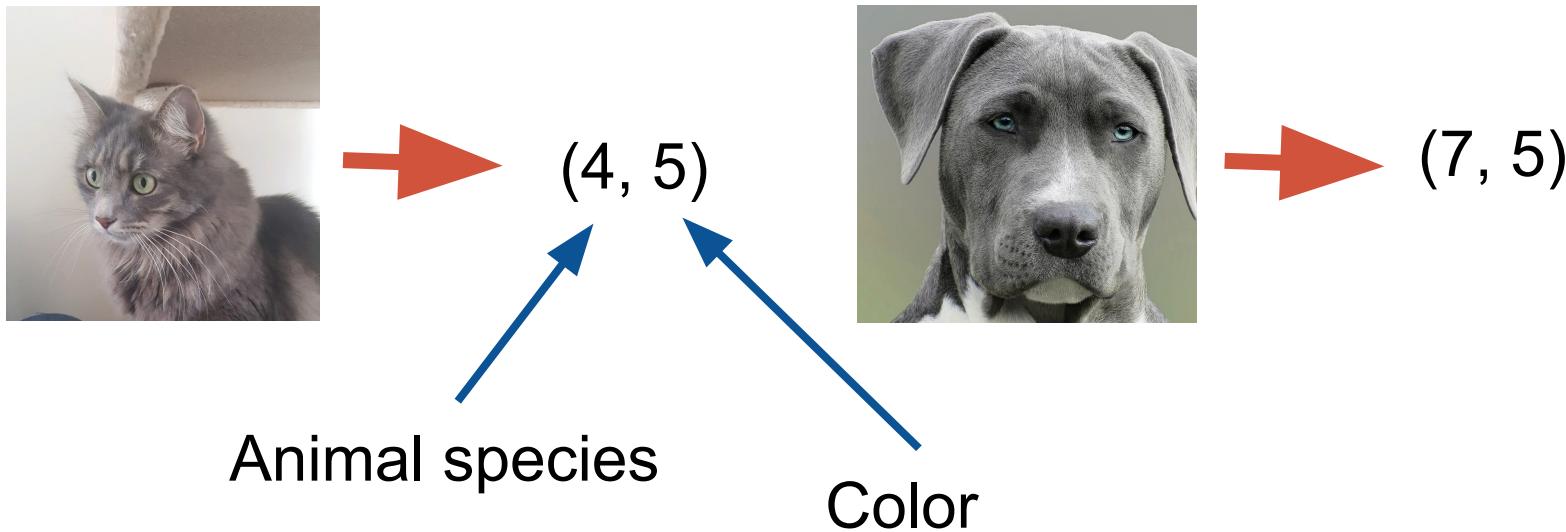
(4.2, 9.1)



(7.9, 4.9)

First Unsupervised Learning Idea

- Ideal magic function f

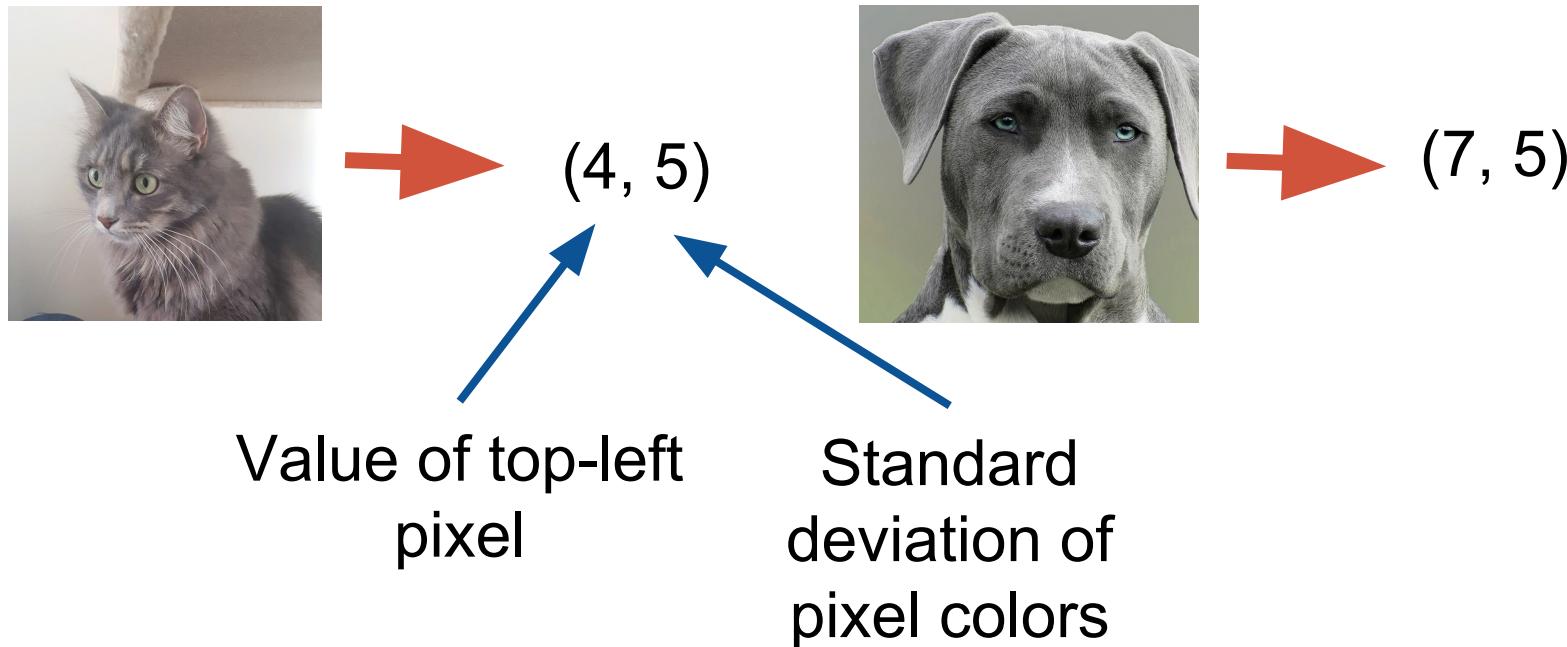


Getting this f would be so nice!

Given f , we can recognize animals with a linear classifier!

First Unsupervised Learning Idea

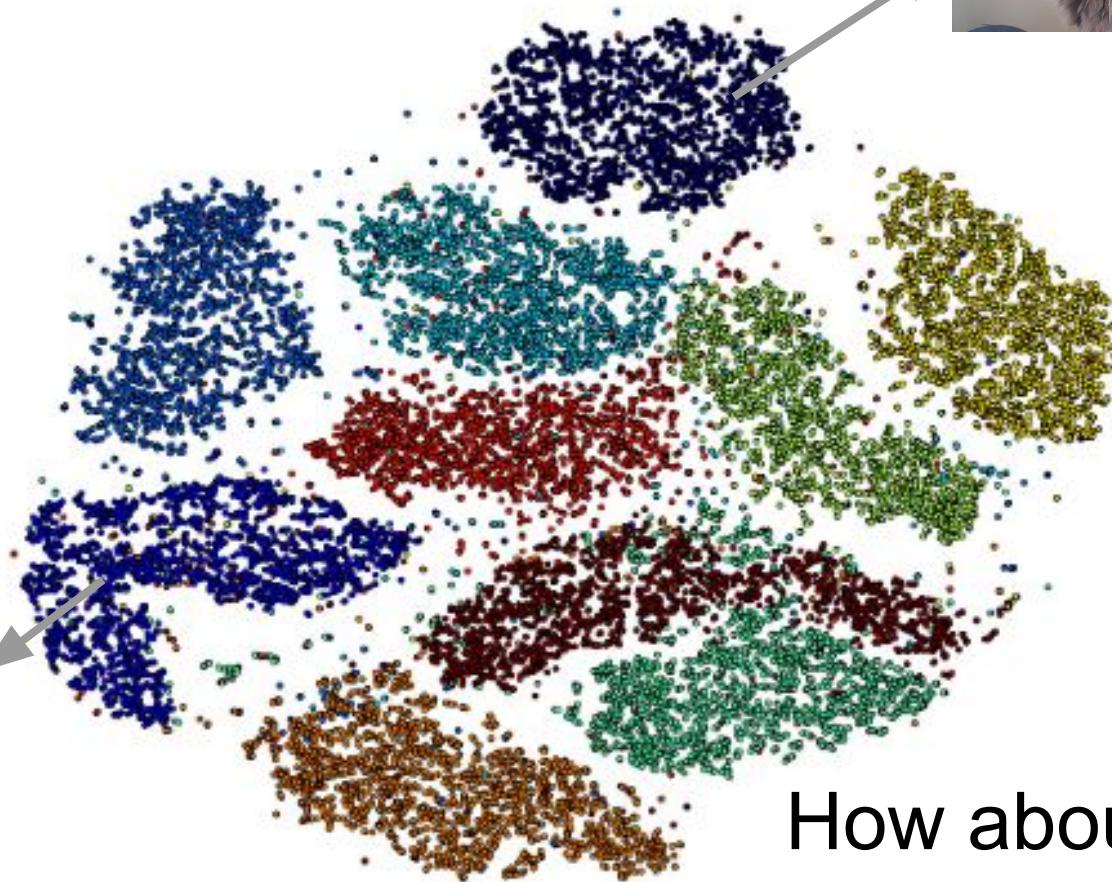
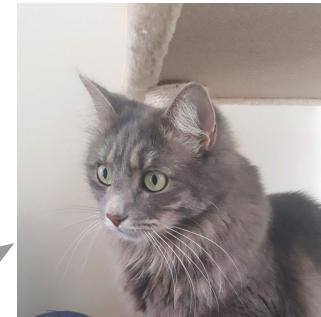
An alternative magic function f



This f isn't so useful

Task: We try to learn the most “useful” $f()$ from unlabeled data.
How to define usefulness? [subjective: Natural, human tasks]

One solution: Clustering



How about K-Mean?
PCA?



How about K-mean with L2?

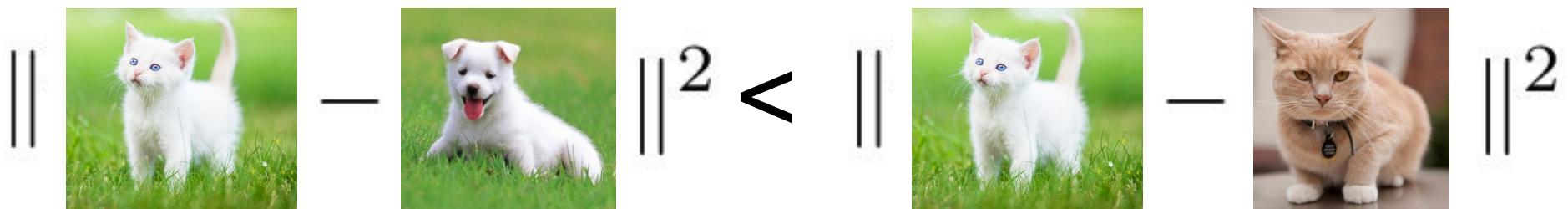
We need a distance metric: How about pixelwise L2 norm?

$$\sum_p |I_p^1 - I_p^2|^2$$

In vector form:



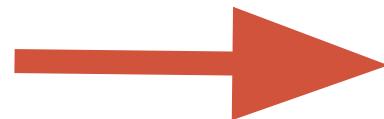
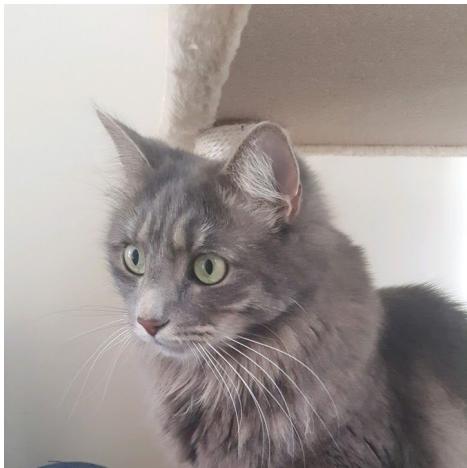
How about K-mean with L2?



White cat is more similar to white dog than to orange cat!

Pixel-wise L2 doesn't work that well. Too low-level comparison.
How can we come up with a useful distance metric? [DNN]

How about PCA?



(4.1, 5.0)

PCA

Dimensionality reduction
from $(h \times w)$ dim to 2 dim

Will this work?

Historically, can be used to do simple face recognition.

PCA: Eigenfaces

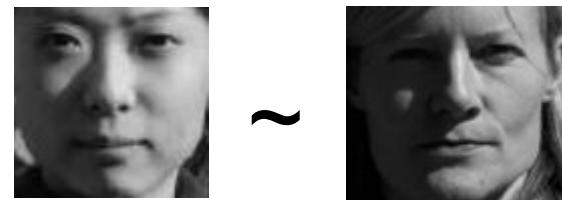


PCA basis

PCA: Eigenfaces

$$\text{Face} = \text{Eigenface}_1 + \text{Eigenface}_2 + \text{Eigenface}_3 + \text{Eigenface}_4 + \text{Eigenface}_5 + \dots$$

$\times 0.3 \quad \boxed{\times 0.2}$ $\times 0.7 \quad \times 0.9 \quad \times 0.1$

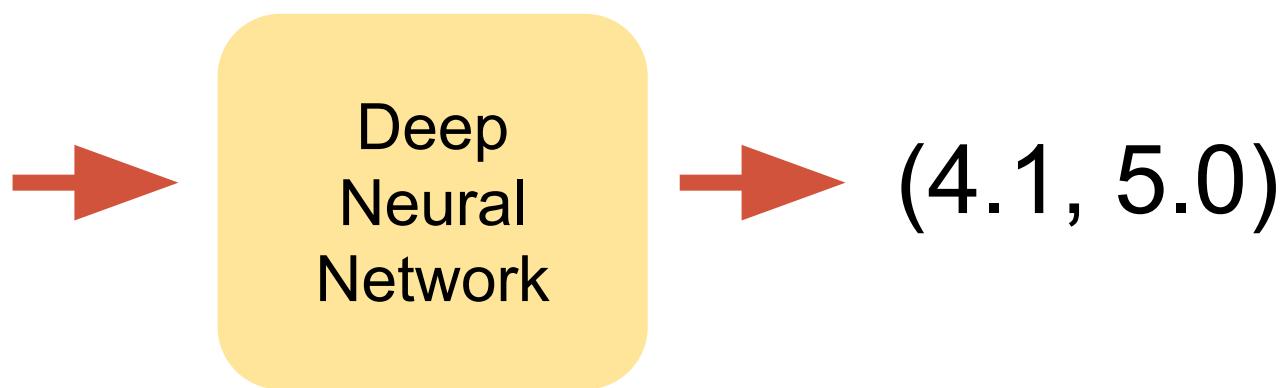
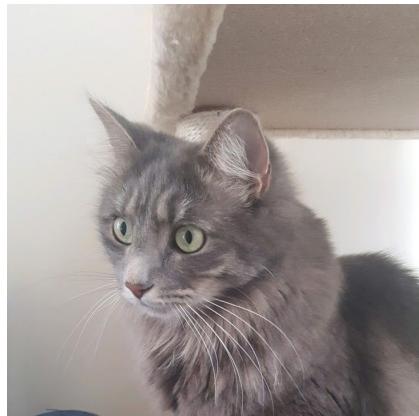


Problems:

1. PCA is a linear operation (as well as LDA or fisherface). Many interesting attributes are highly non-linear, e.g., identity, emotion, and cannot be factored linearly.
2. Faces have to be aligned and centered for this to work
3. Illumination dominates!

How about DNN?

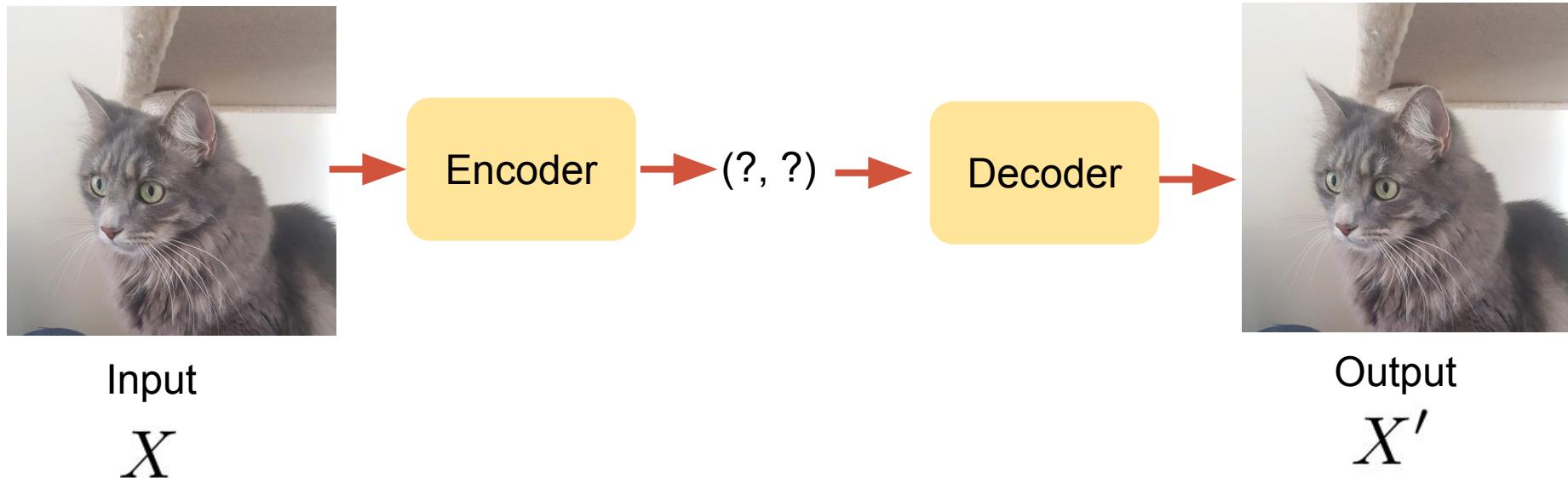
We know that DNN can model highly complex mapping.
How about modeling f with a neural net?



But how to train it? We don't know the target, groundtruth
two numbers.

Autoencoder

Auto means self!



Input

X

Output

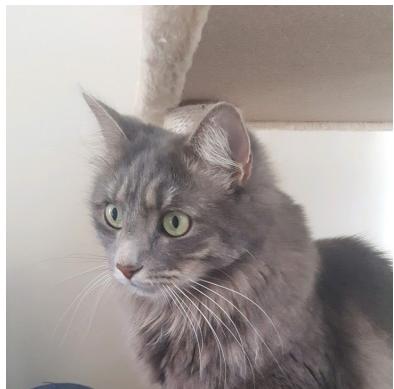
X'

Now we can train the network with L2 Loss:

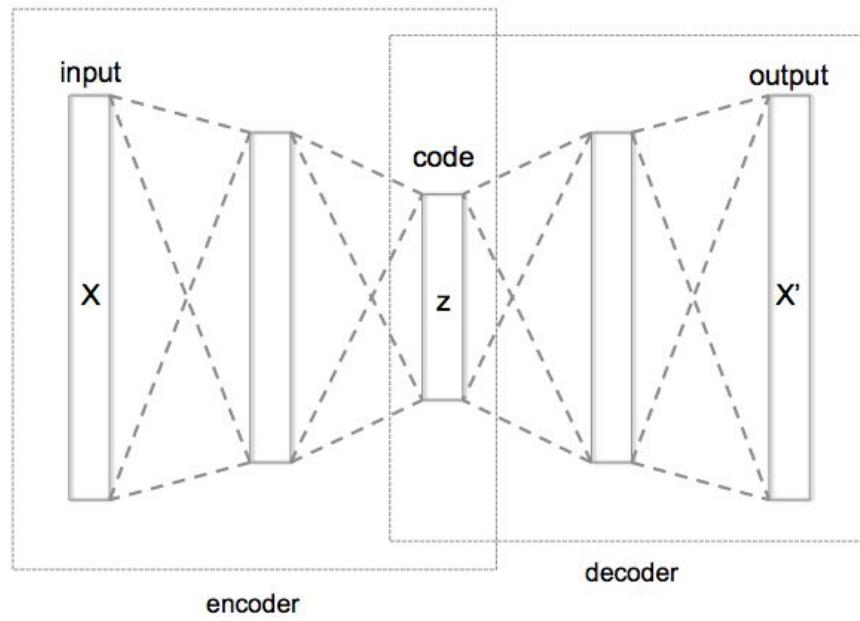
$$\|X - X'\|^2$$

Autoencoder

Auto means self!



Input
 X

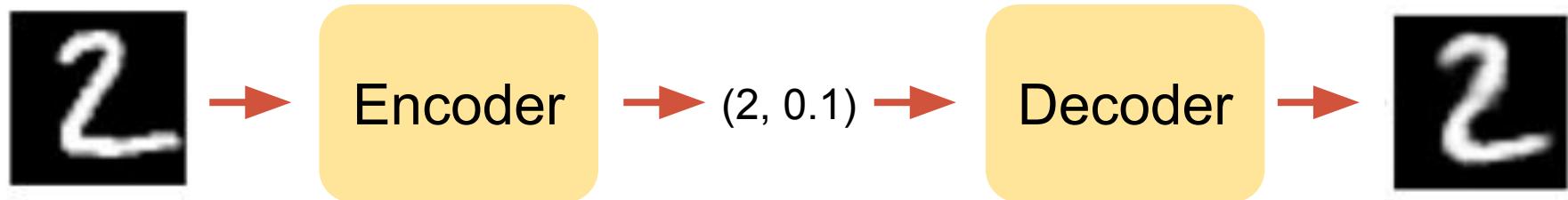


Output
 X'

Now we can train the network with L2 Loss:

$$\|X - X'\|^2$$

Autoencoding Game



Both Encoder and Decoder see the entire dataset.

Encoder: What two numbers should I send you so that you can re-draw my input as accurate as possible?

Encoder: Maybe the first number would directly correspond to what digit it is. The second, perhaps, the thickness of the stroke?

Turns out DNN does surprisingly well at encoding important attributes.

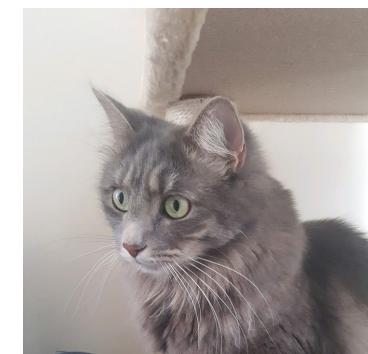
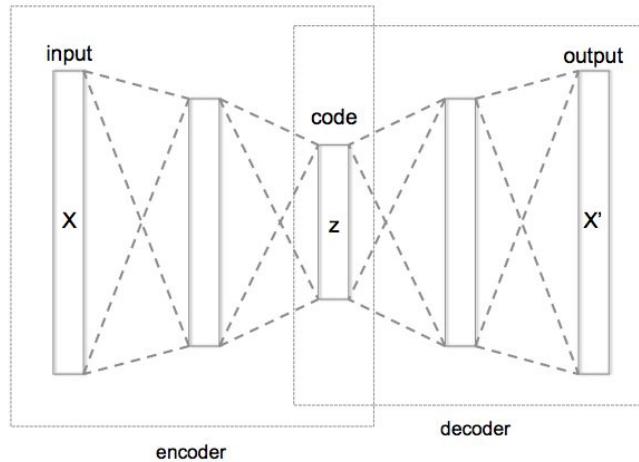
Autoencoder

Can we pick an arbitrary dimension for the code?

- When z has the same dimension as the input and output, NN cheats and outputs an identity mapping \rightarrow useless.



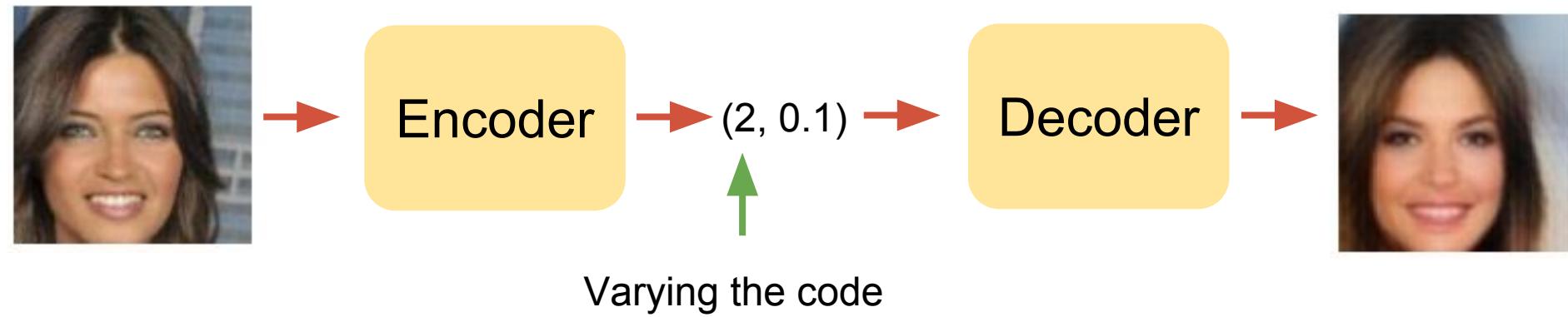
Input
 X



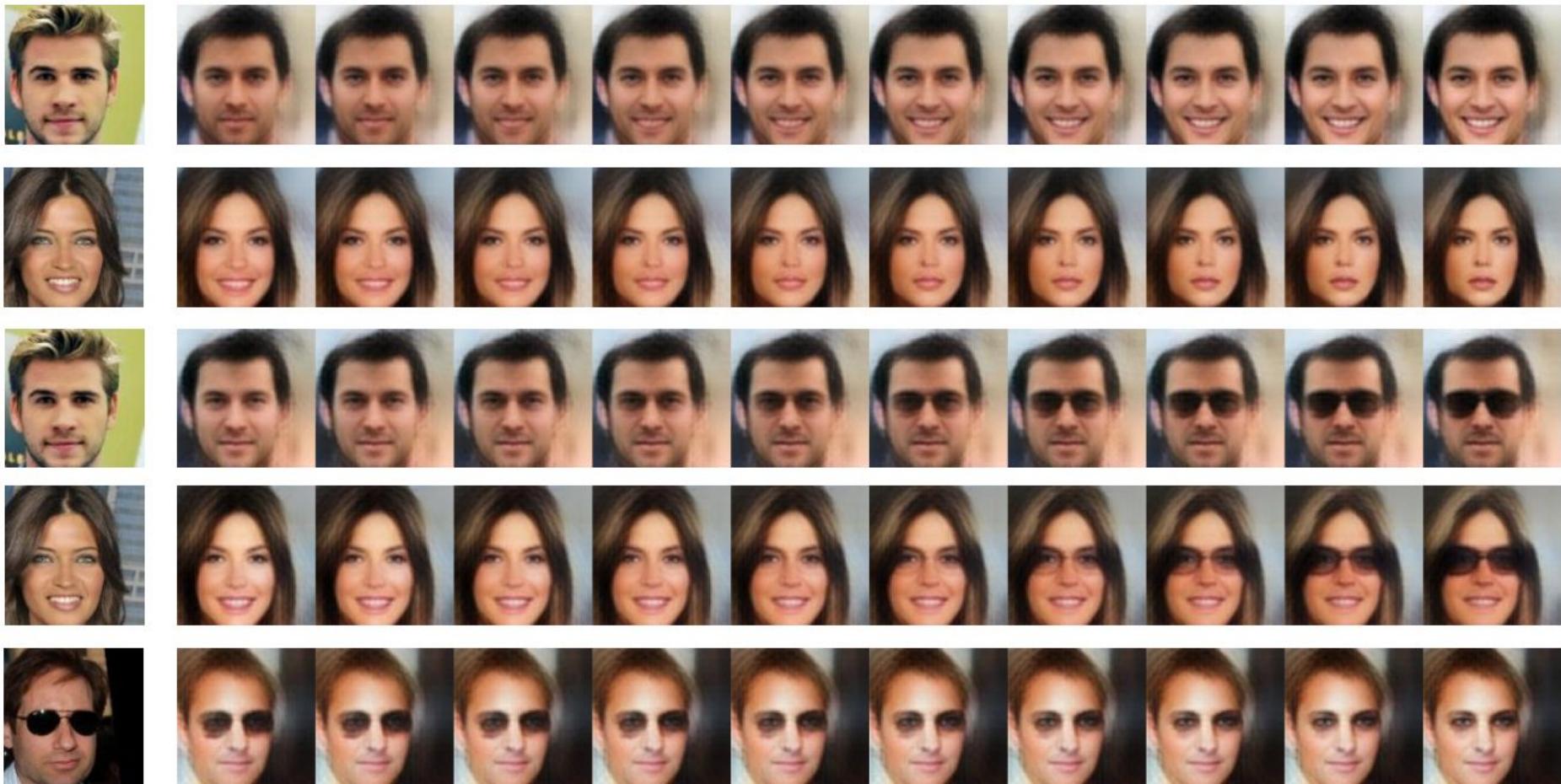
Output
 X'

- When z dim \ll input dim, NN has to work hard!

Inspecting attributes learned from AE



Attributes from autoencoder

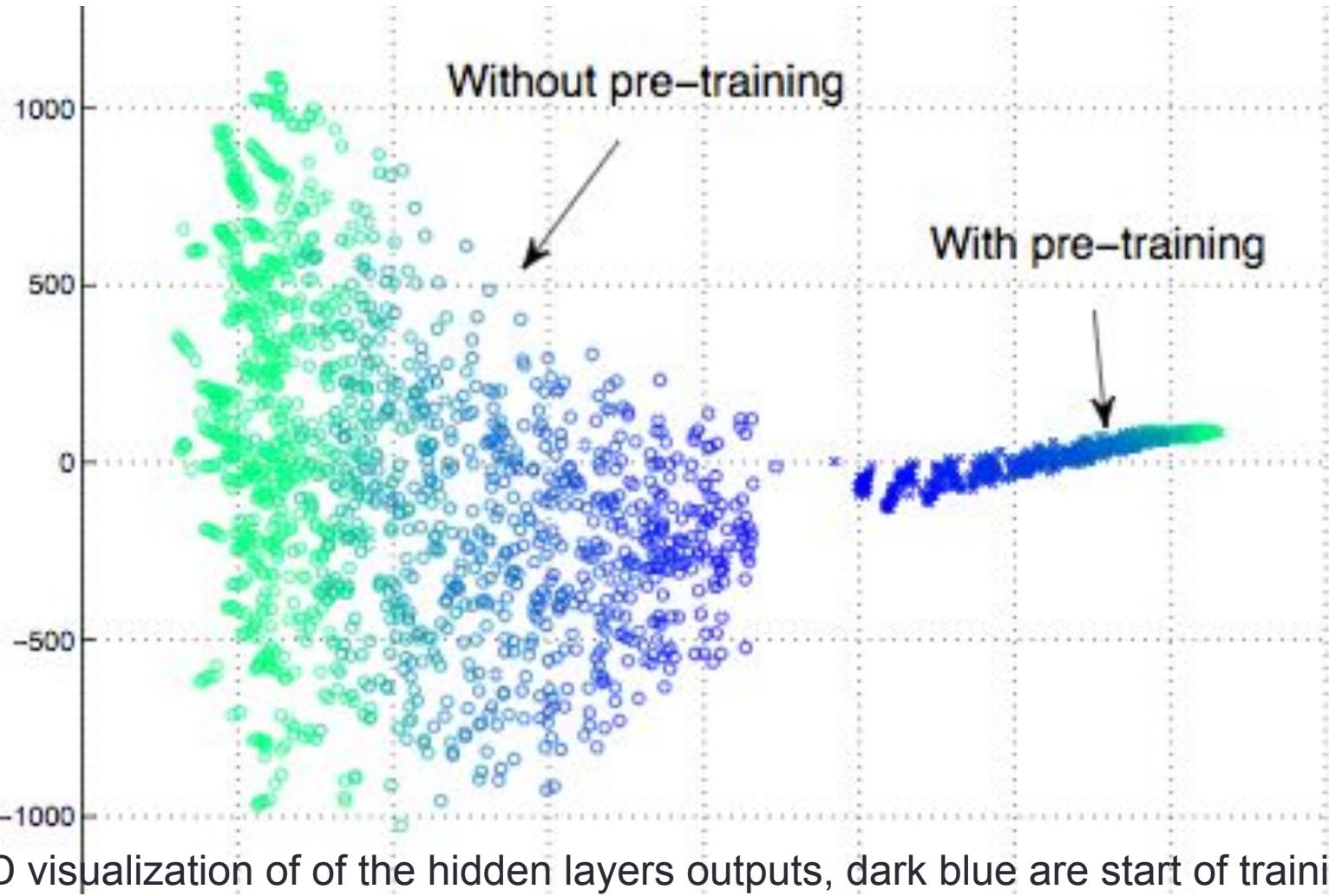


Utilizing autoencoder for supervised tasks

Many ways to help with a supervised task such as recognition.

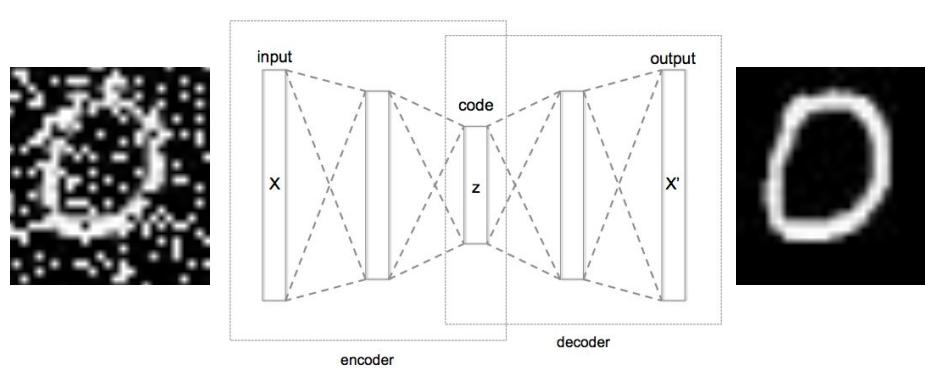
1. Append the input with the code from the encoder
2. Stick a classifier on top of the encoder
3. Used for pretraining a network

Pre-training effects



Denoising autoencoder

- Motivation: Making AE more robust -- really learn what's important
- An autoencoder that denoise corrupted inputs
 - Blur image, sound corrupted by noise
- How?
 - Corrupt your input $x \rightarrow x''$
 - Use x'' as the input, and minimize the same loss
 - Loss = $\|x-x'\|^2$



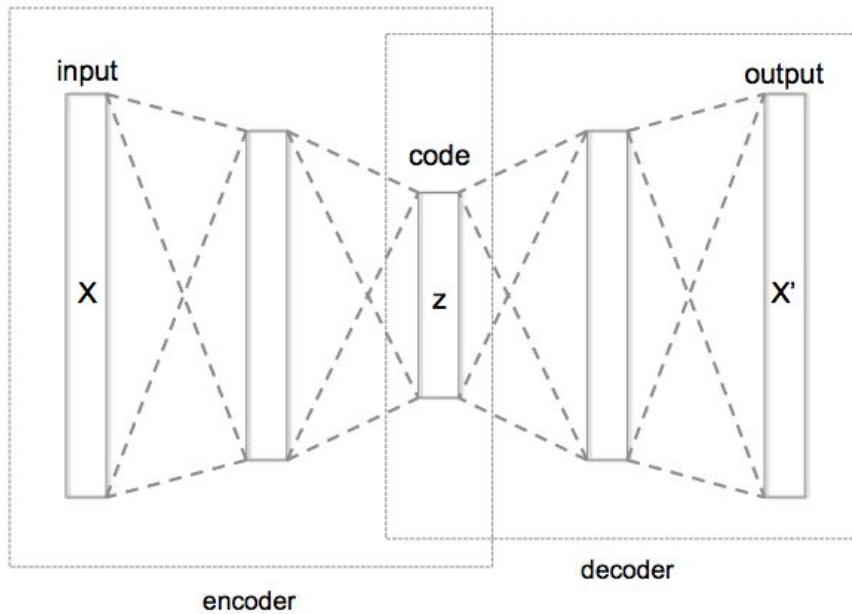
Example

Original	Corrupted	Generated
7 2 1 0 4	7 2 1 0 4	7 2 1 0 4
1 4 9 5 9	1 4 9 5 9	1 4 9 5 9
0 6 9 0 1	0 6 9 0 1	0 6 9 0 1
5 9 7 3 4	5 9 7 3 4	5 9 7 3 4
9 6 4 5 4	9 6 4 5 4	9 6 4 5 4

Other uses

- Besides denoising, can also be used to make your “code” more robust to the type of corruption you introduced.
- Finding similar images. Combining with clustering algos.
- Generative models / generating photos

Autoencoders



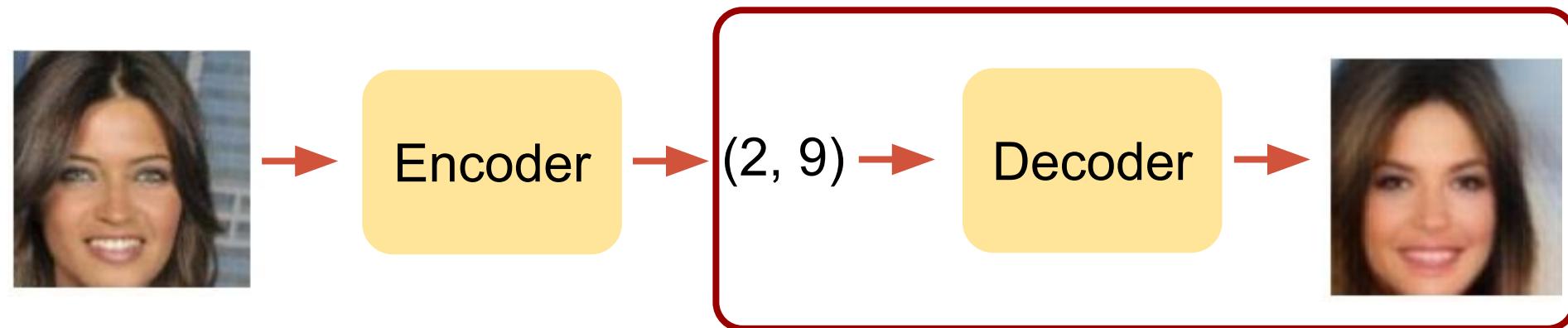
- Automatically learn features
- Non-linear dimensionality reduction / data compression

Variational Autoencoder

Original and by-product motivations:

1. Allows approximation of intractable posterior.
2. Generate photos from the decoder

Motivation #2 -- Generating Photos



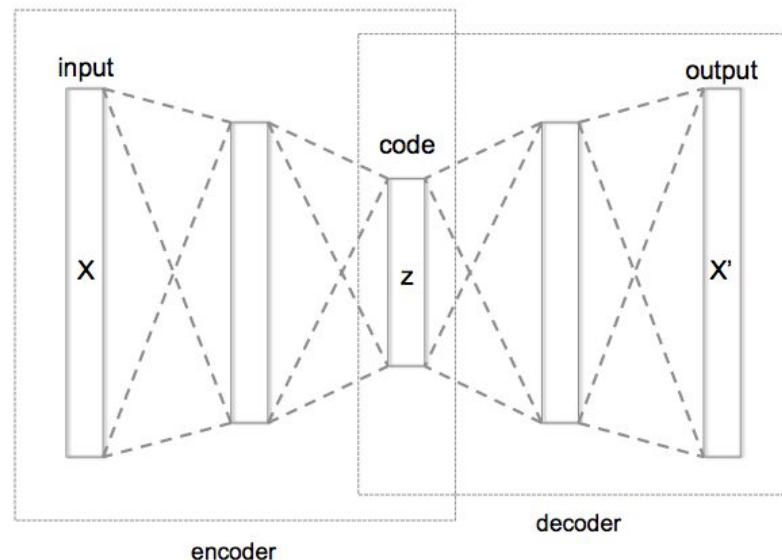
Use this part to generate an image from random code.

But how do you generate the two numbers? Uniform? Gaussian?
Encoder could output two numbers with some crazy distribution.

VAE allows us to force the latent code to follow some distribution like a unit normal distribution.

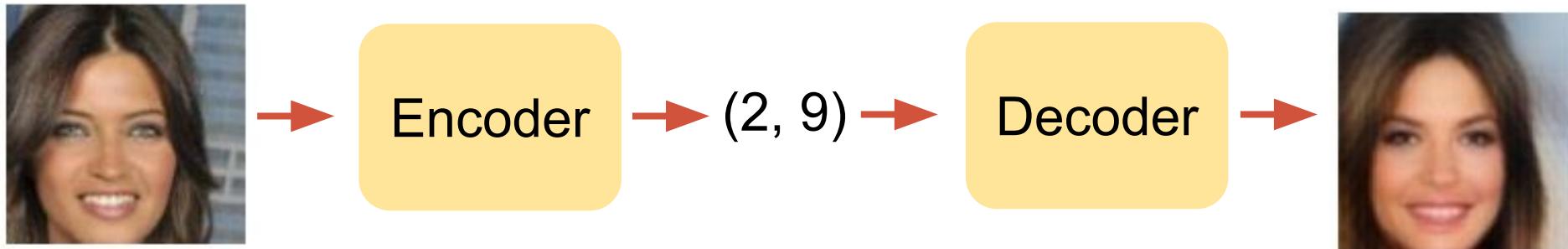
Motivation #1 -- Solving Intractable Inference

Autoencoder can be viewed from a probabilistic perspective.

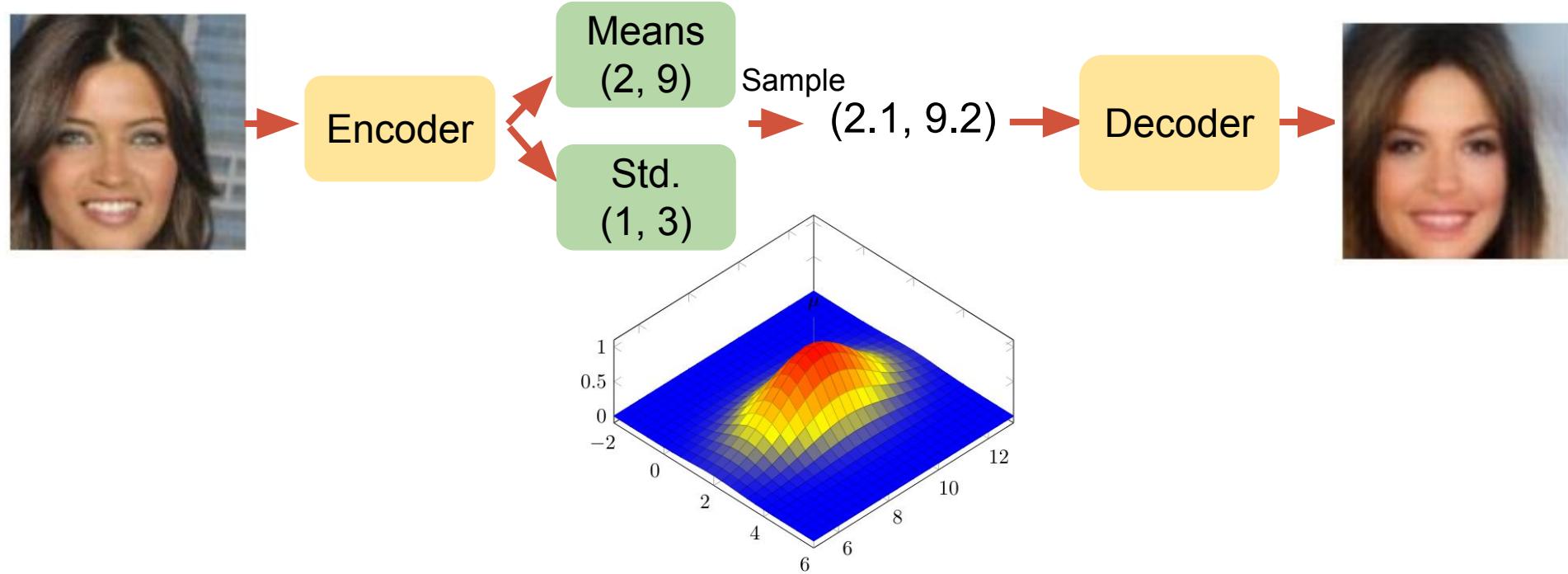


However, a standard AE isn't quite "probabilistic."
I.e., the encoder deterministically outputs z given x . Rather we want a distribution of the code (posterior $p(z|x)$).

Standard AE



Probabilistic AE



Autoencoding Variational Bayes: Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x})$$

x Observed data

z Latent variable
(hidden code)

Bayesian Review

One important inference problem : Posterior inference

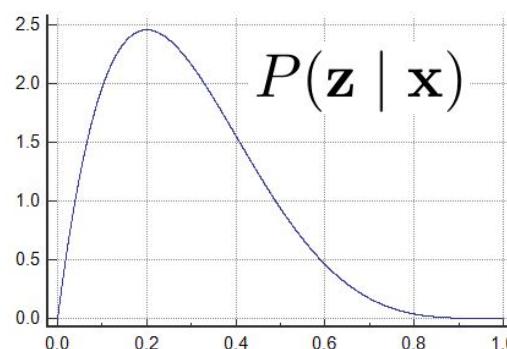
$$p(\mathbf{z} \mid \mathbf{x})$$

x Observed data

2 heads out of 10 coin flips
(Binomial Dist)

z Latent representation
(hidden code)

The coin fairness
 $z = 0.5$, fair coin
 $z = 1$, always head
 $z = 0$, always tail



Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x})$$

x Observed data



z Latent representation
(hidden code)

Class label cat, color,
can fly?, ...

$$p(\mathbf{z} \mid \mathbf{x})$$

Tells you **not only** what animal is the most likely,
but also how “orange cat” it is, how “white dog” it is, etc --
the probability of all hidden code

Variational Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{z}, \mathbf{x})}{\boxed{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}}$$

Hard to compute because \mathbf{z} can be high-dimensional

E.g. $\mathbf{z} = \{z_1, z_2, z_3, \dots\}$

z_1 : class label {cat, dog, rabbit, rat}

z_2 : color, a real number

z_3 : ability to fly {yes, no}

$$\int d\mathbf{z} = \sum_{z_1} \int_{z_2} \sum_{z_3} \dots dz_2$$

Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{z}, \mathbf{x})}{\boxed{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}}$$

A solution:

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$

with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$

and solve this as an optimization problem.

Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$
with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$
and solve this as an optimization problem.

Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$ with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$ and solve this as an optimization problem.

Best $q^*(\mathbf{z} \mid \mathbf{x})$ is the one closest to $p(\mathbf{z} \mid \mathbf{x})$

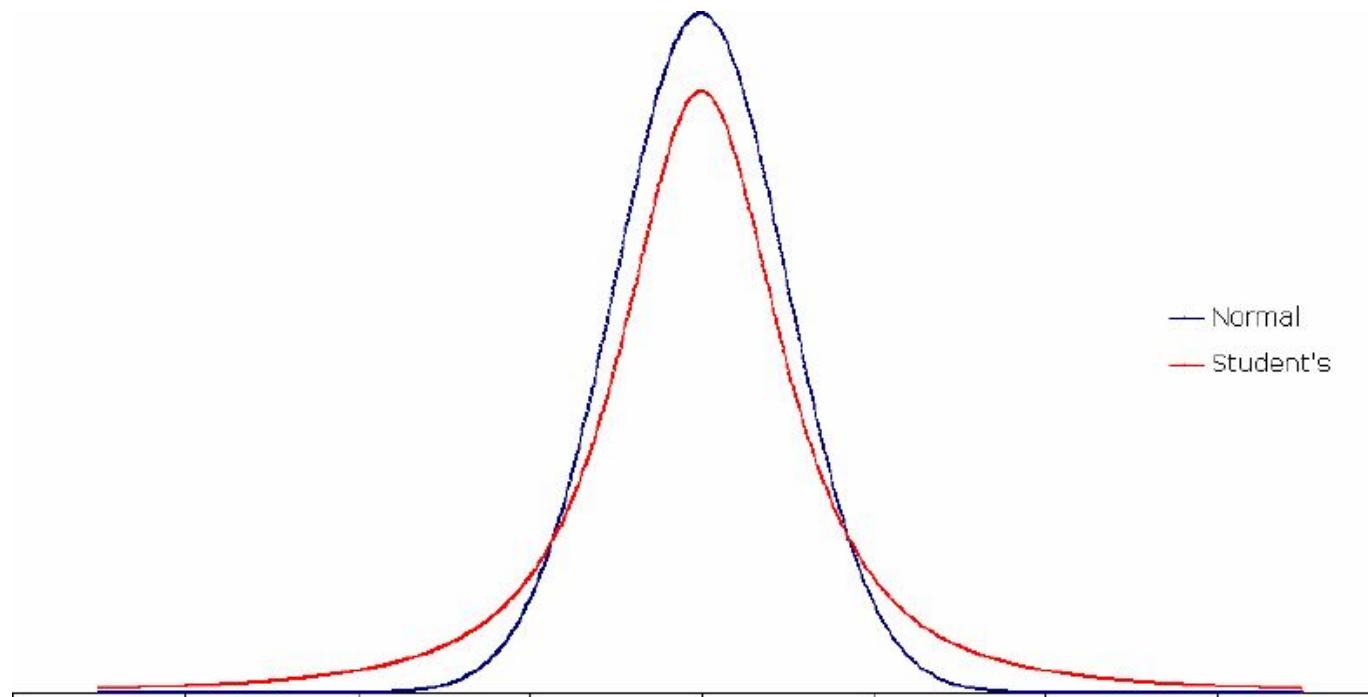
A solution: $q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname{argmin}_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})]$

- But why not $\text{KL}[p(\mathbf{z} \mid \mathbf{x}) \parallel q(\mathbf{z} \mid \mathbf{x})]$ or $D(p(\mathbf{z} \mid \mathbf{x}), q(\mathbf{z} \mid \mathbf{x}))$?
- How is this related to autoencoder?

Digression: Model evidence

Model evidence (or marginal likelihood) is simply $p(\mathbf{x})$

$$P(\mathbf{x}) = \text{Int } P(\mathbf{x}, \mathbf{z}) d\mathbf{z}..$$



Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$ with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$ and solve this as an optimization problem.

Because doing this (i.e., using $\text{KL}(q \parallel p)$):

$$q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname{argmin}_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})]$$

Corresponds to maximizing the evidence lower bound (ELBO)!

Proof

This KL is hard to compute



$$q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname{argmin}_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})]$$

$$\text{KL}[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] = \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{z} \mid \mathbf{x})]$$

(by definition)

$$= \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x})$$

(expanding $P(z \mid x)$, $\mathbb{E}_q[\log p(x)] = \log p(x)$)

$$= \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x})$$

Constant w.r.t q. So, we can safely kill it

$$\min_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] \iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

Questions: 1. What's this new term? 2. Is it easier to evaluate?

Proof

$$\min_q \text{KL} [q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] \iff \min_q \mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

Let's interpret this new term:

From previous slide, we have:

$$\log p(\mathbf{x}) = \text{KL} [q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

We know $\text{KL} \geq 0$, always!

$$\log p(\mathbf{x}) \geq \boxed{-\mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]}$$

(Lower bound of model evidence)

$$\begin{aligned} \min_q \mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] &\iff \max_q -\mathbb{E}_q [\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] \\ &\iff \max_q \text{ELBO} \end{aligned}$$

But how to compute this ELBO term then?

Maximizing ELBO

$$q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname{argmin}_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})]$$


$$\max_q \text{ELBO} = \max_q -\mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

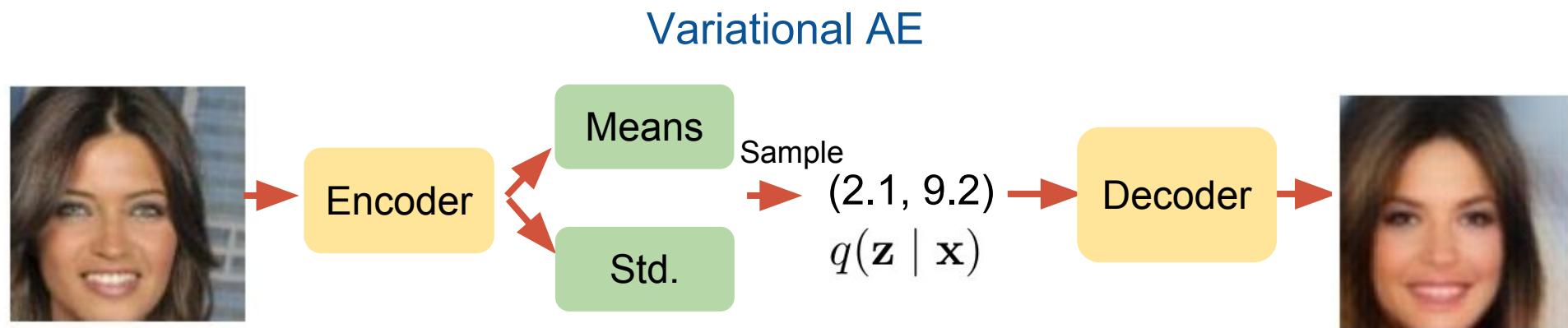
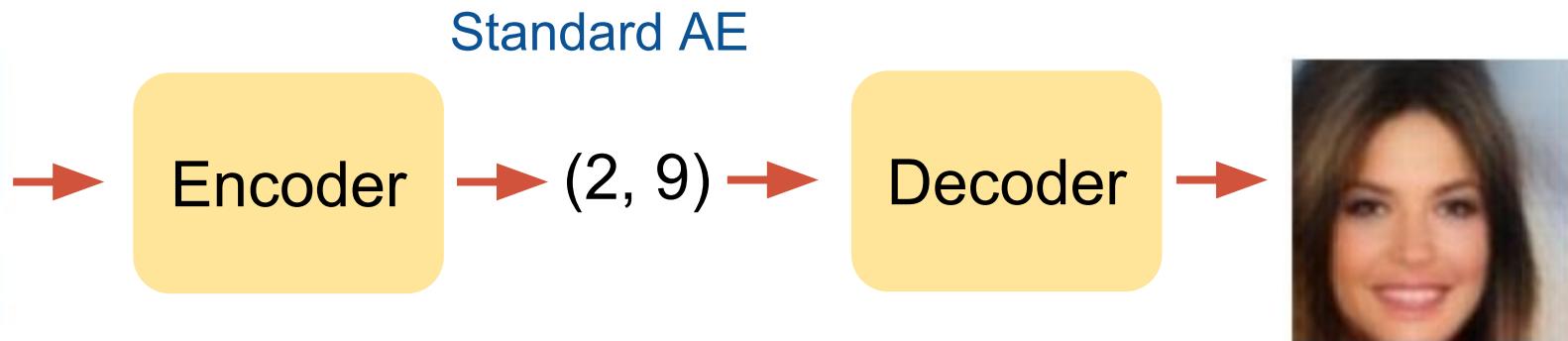
$$\iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

$$\iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{z})]$$

$$\iff \min_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

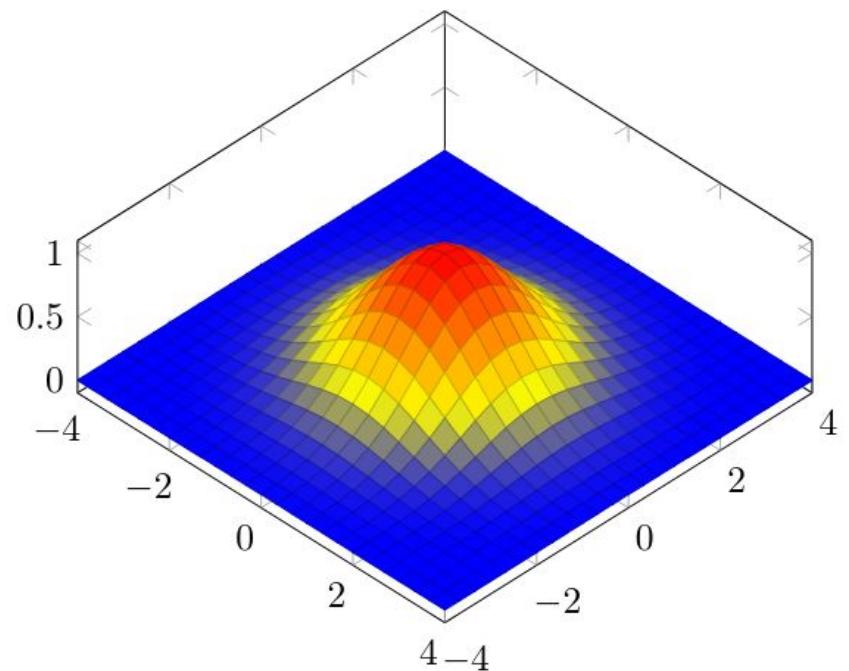
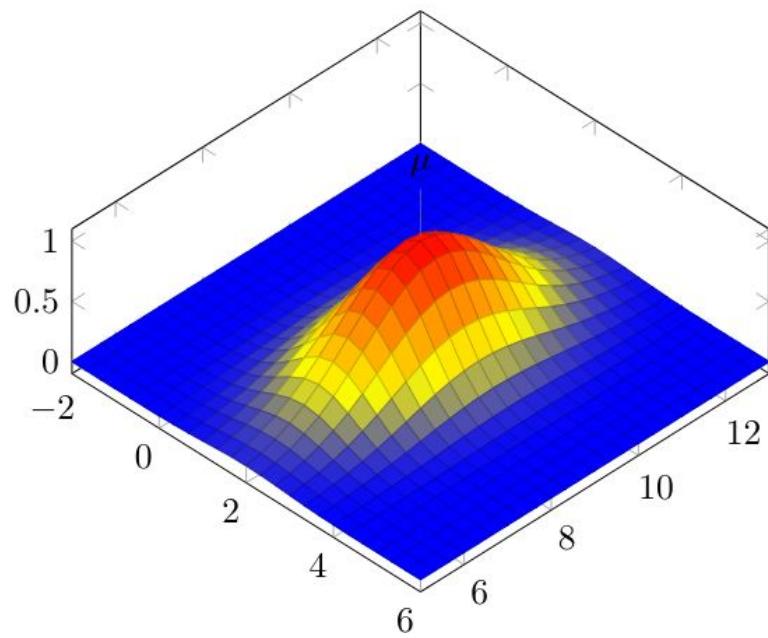
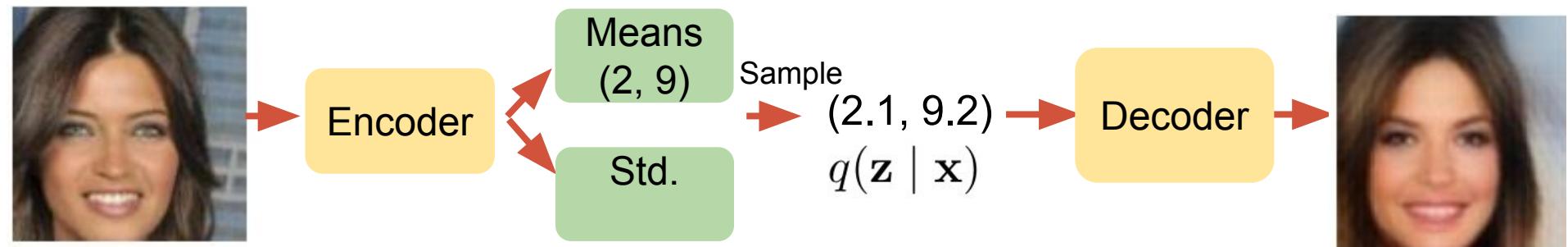


We're free to pick



*Output as a sample
from $P(x | z)$.

$$\min_q \text{KL}[q(\mathbf{z} | \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} | \mathbf{z})]$$



$q(\mathbf{z} \mid \mathbf{x})$ is forced to be close to $p(\mathbf{z}) = \mathcal{N}(0, I)$

$$\min_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

What is this second term?

Second Term: Data Log Likelihood

$$\min_q \text{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

If we also model $p(\mathbf{x} \mid \mathbf{z})$ as a Gaussian distribution:

$$p(\mathbf{x} \mid \mathbf{z}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-u)^2}{2\sigma^2}}$$

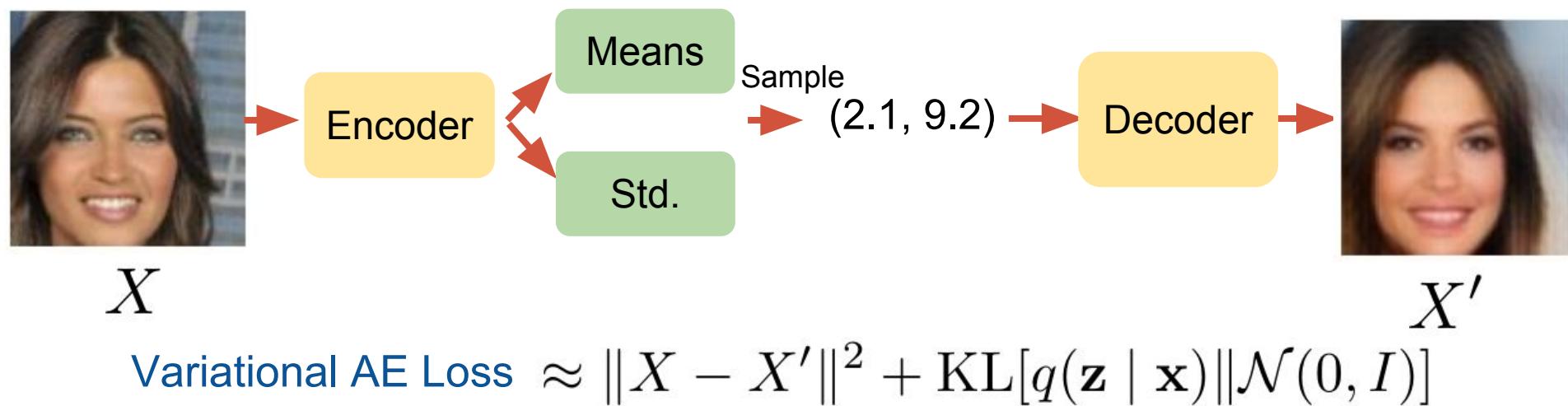
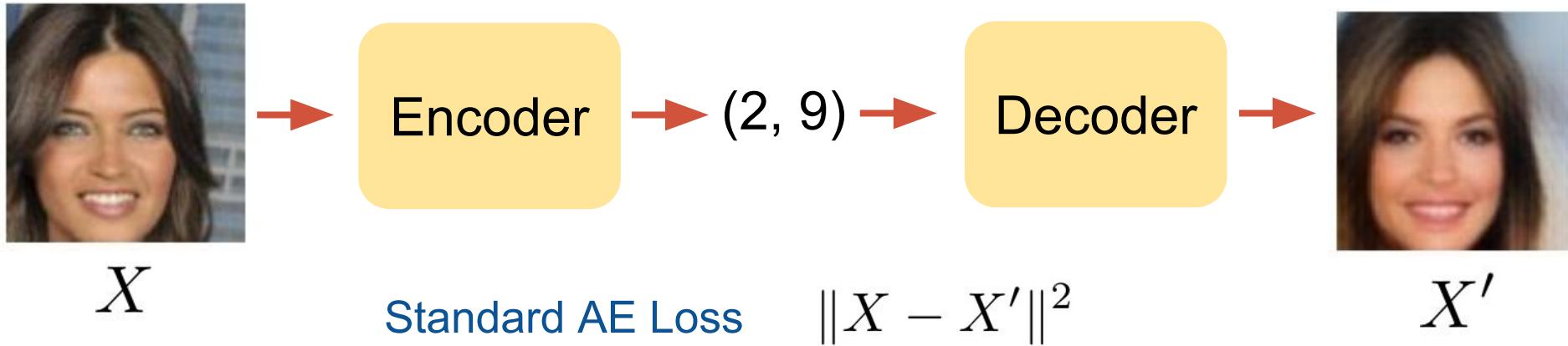
(Let's consider a single pixel case
u is the output from the decoder
x is the ground-truth pixel)

$$\propto \exp(-(x-u)^2)$$

$$-\log[\exp(-(x-u)^2)] = (x-u)^2 \quad (\text{with constants hidden})$$

Becomes the standard L2 error

Yet another reason why assuming Gaussian is nice



Precisely,

$$\min_{q_\phi(\mathbf{z} \mid \mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z} \mid \mathbf{x})} [\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

Last Problem

$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

When we need to compute the gradient with respect to the network parameters (I.e., ϕ, θ) for SGD, we're struck because we can't back-prop through the sampling for computing $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}$

Example:

$$z = y^2 \quad z \sim \mathcal{N}(y, 1) \quad z \sim q_\phi(z \mid x)$$

$$\frac{dz}{dy} = 2y \quad \frac{dz}{dy} = ?? \quad \frac{dz}{d\phi} = ??$$

Reparametrization Trick

$$z \sim \mathcal{N}(\mu, \sigma)$$

$$z = \epsilon_i \times \sigma + \mu$$

$$\frac{\partial z}{\partial \mu} = ??$$

$$\epsilon \sim \mathcal{N}(0, 1)$$

$$\frac{\partial z}{\partial \sigma} = ??$$

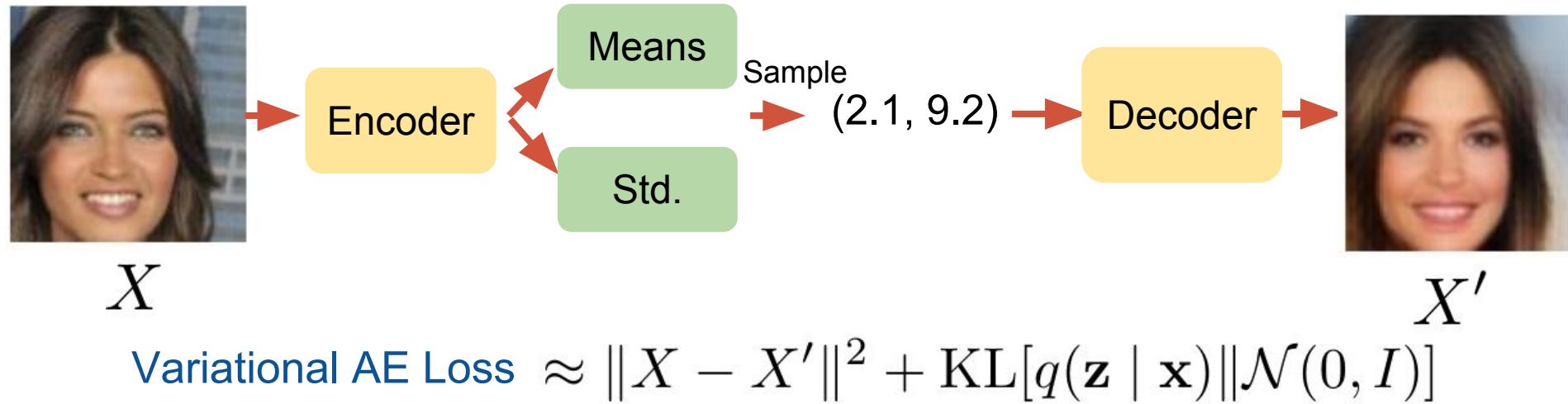
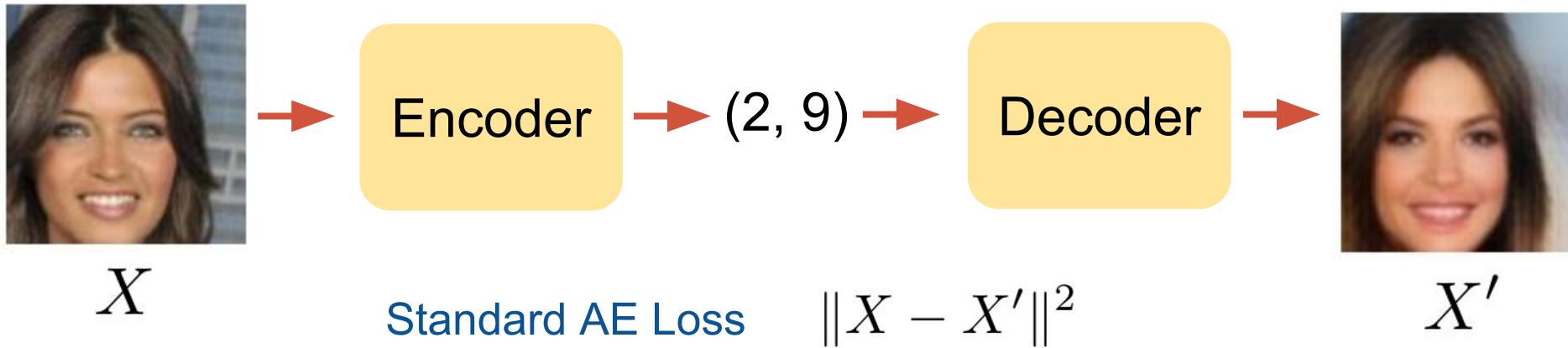
$$\frac{\partial z}{\partial \mu} = 1$$

$$\frac{\partial z}{\partial \sigma} = \epsilon_i$$

$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

$$\iff \min_{q_\phi(\mathbf{z}|\mathbf{x})} \text{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{p(\epsilon)}[\log p_\theta(\mathbf{x} \mid \hat{\mathbf{z}})]$$

$$\hat{\mathbf{z}} = \epsilon \cdot \sigma + \mu, \quad p(\epsilon) \sim \mathcal{N}(0, I)$$



Disentangling Representation: β -VAE

We want compact meaningful representation.

But also disentangled representation

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

Skin color

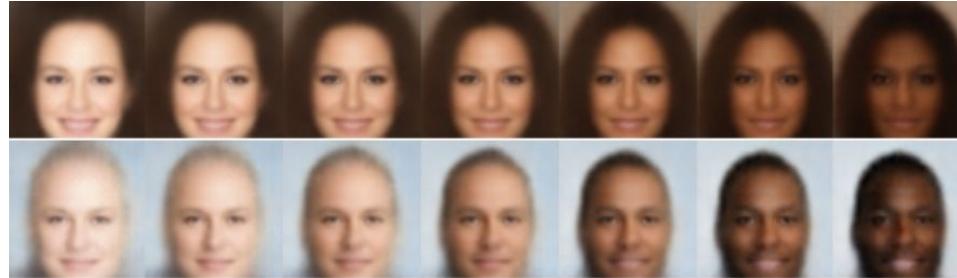
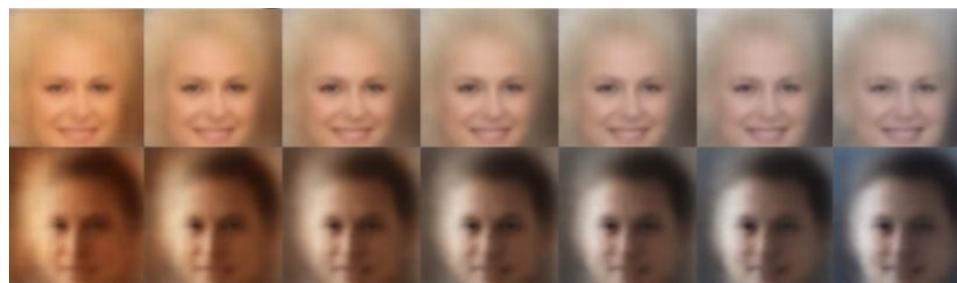
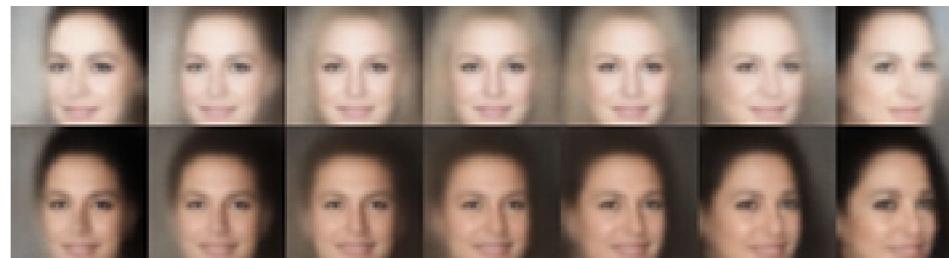


Image
Saturation



Head
Angle



Age / Gender



Variational Autoencoders (VAE)

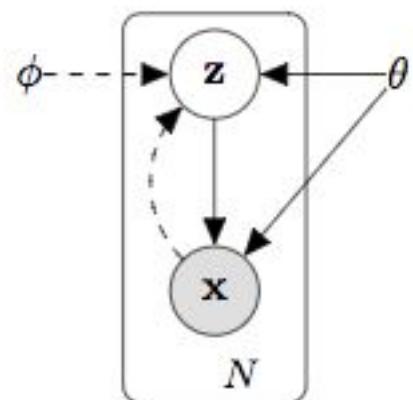
- Assume X is generated from a latent variable Z
 - Ex: a picture of a cat is generated from class label cat
- VAE wants to find this latent variable Z in the coding layer
- Our code can be generated from
 - $q_\phi(z|x)$ (our neural network)
 - which tries to mimic the true $p_\theta(z|x)$ that we don't know.

VAE reward (not loss)

$$E[\log P(X|z)] - D_{KL}[Q(z|X) \| P(z)]$$

Decoder

Encoder True Code



VAE reward explained

- We want to maximize

$$E[\log P(X|z)] - D_{KL}[Q(z|X) \| P(z)]$$

- D_{KL} is the KL-divergence
 - Distance between distributions
$$D_{KL}(P\|Q) = - \sum_i P(i) \log \frac{Q(i)}{P(i)},$$
- The first term means we need to generate X in the best way possible from z. This is actually maximum likelihood estimation.
- What about $P(z)$?
 - This is what we get to pick, and the VAE will try to match it.
 - Simplest choice is $N(0,1)$
- For more info, read <https://arxiv.org/abs/1606.05908>

And <https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/>

Generating Images

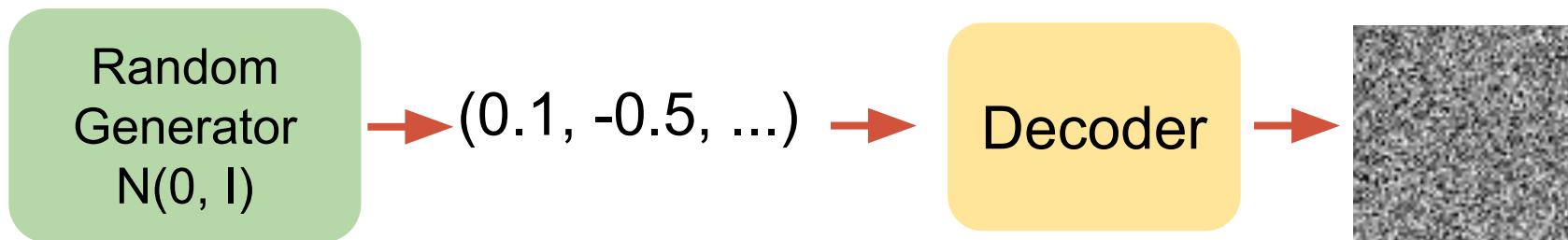
VAE can be used to generate images by feeding numbers into the decoder.

Are there other ways?

Given a collection of, say, faces, how can we generate new faces?

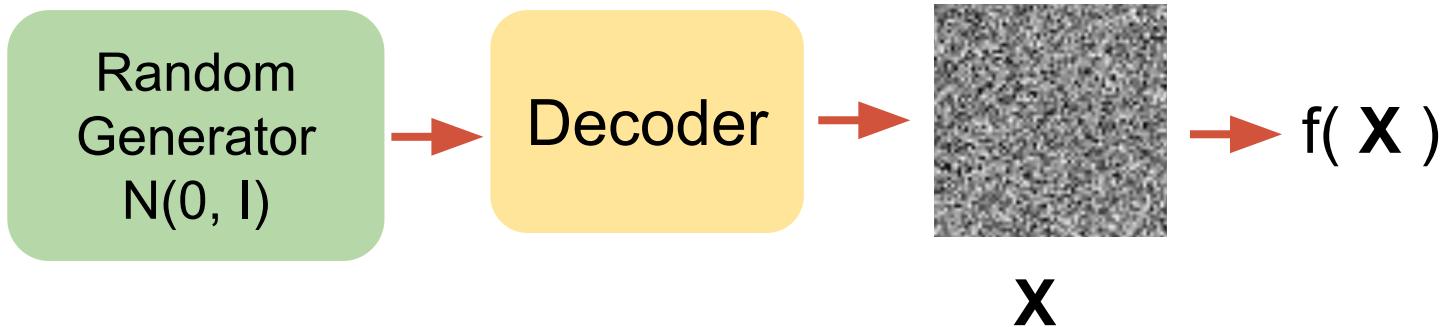
Maybe let's start with a decoder, because we need something that outputs an image anyway.

Generating Realistic Faces



What kind of loss should we use?

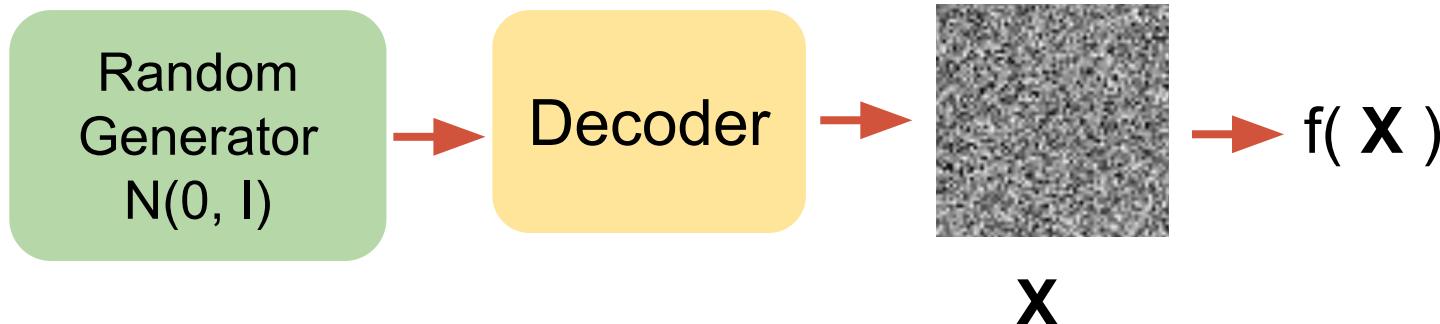
Generating Realistic Faces



We'll be done if we can come up with a smooth function f , that takes in the output image x and outputs $[0, 1]$ where:

$$f(\text{Image of Face}) = 1 \quad f(\text{Image of Non-Face}) = 0.2 \quad f(\text{Image of Noise}) = 0$$

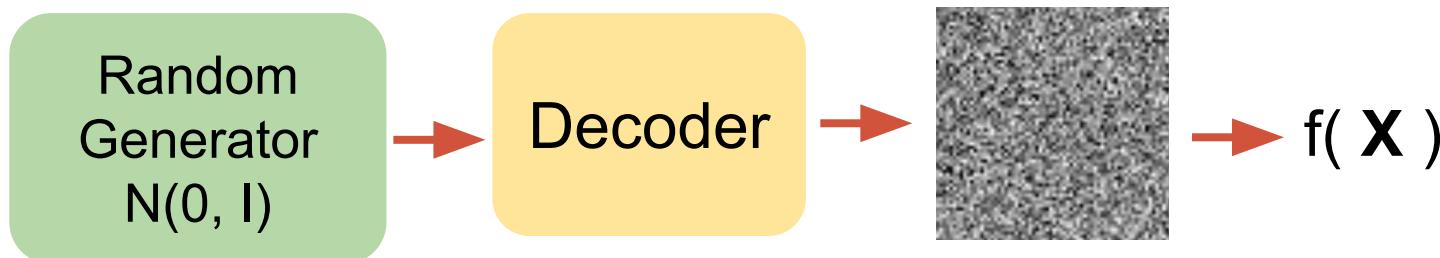
Generating Realistic Faces



$$f(\text{Image of Face}) = 1 \quad f(\text{Image of Tap}) = 0.2 \quad f(\text{Image of Noise}) = 0$$

Given this f , we can compute its gradient with respect to the decoder parameters and use SGD to train. Done!

Generating Realistic Faces



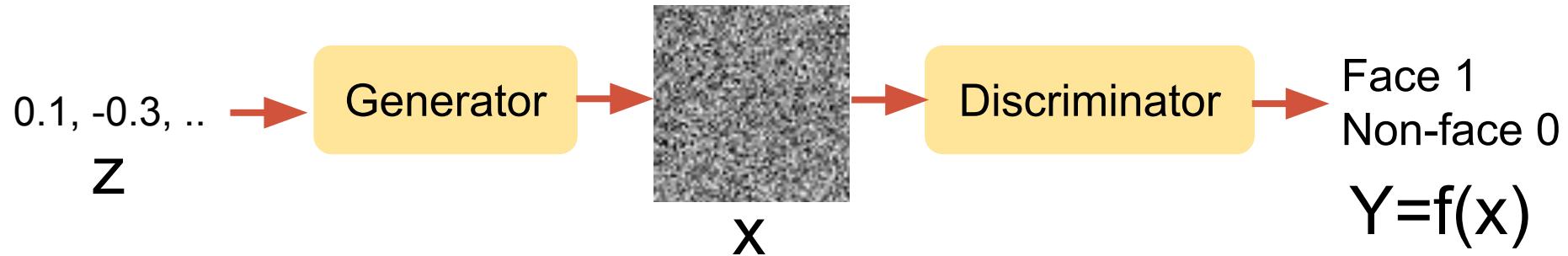
Solution: We will use another network as f . This network will predict the probability of x being a face.

Training this f is a simple supervised learning!

All face photos -> positive examples

All non-face photos -> negative examples

Generating Realistic Faces

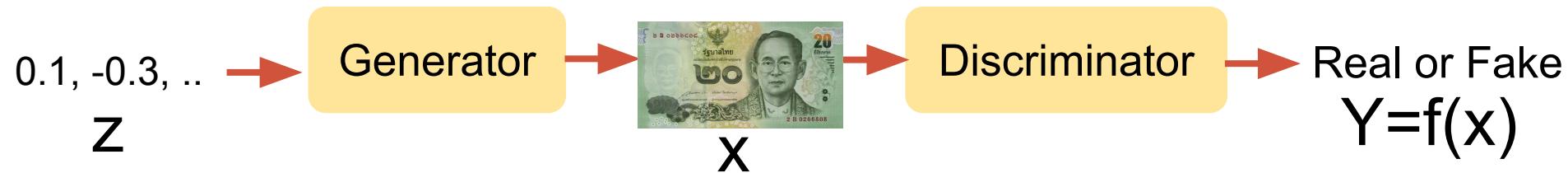


Generative Adversarial Networks (GAN)



- Consider a money counterfeiter
 - He wants to make fake money that looks real
 - There's a police that tries to differentiate fake and real money.
- The counterfeiter is the **adversary** and is **generating** fake inputs. – Generator network
- The police is try to discriminate between fake and real inputs. – Discriminator network

Generative Adversarial Networks (GAN)



- Generator (Money Faker):

- Maximize Y

$$\min_G \mathbb{E}_{\mathbf{z}} [\log(1 - D(G(\mathbf{z})))]$$

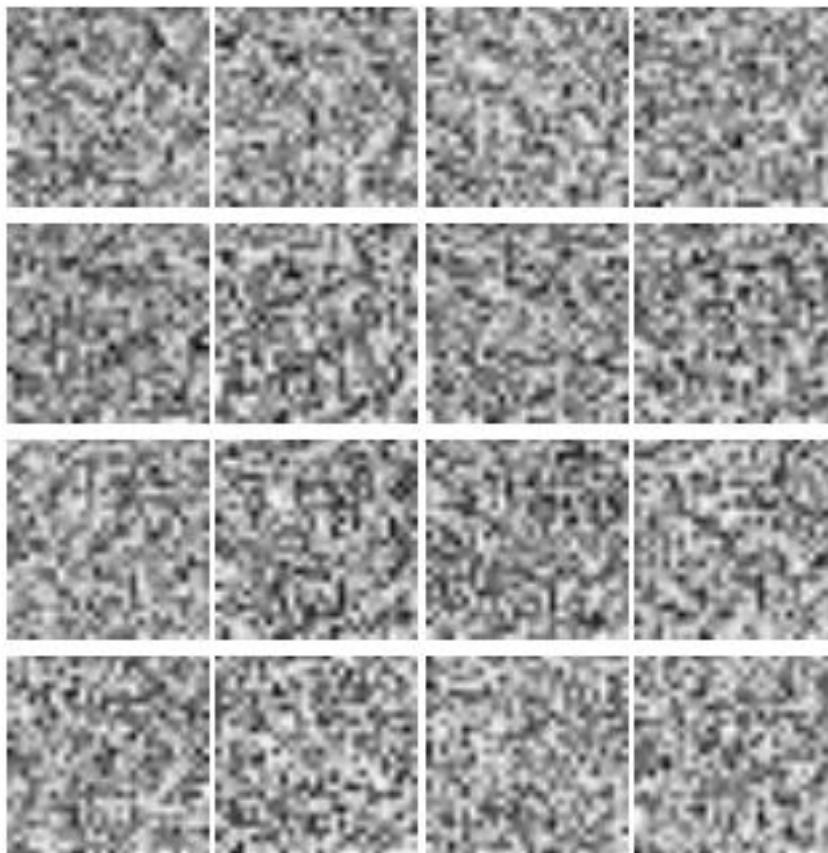
- Discriminator (Police):

- For real images => Maximize Y
 - For generated images from the faker => Minimize Y

$$\max_D \mathbb{E}_{\mathbf{z}} [\log(1 - D(G(\mathbf{z})))] + \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log(D(\mathbf{x}))]$$

GAN example

Generator output starts from random noise and gets better as we train.



Progressive GANS



Progressive Growing of GANs for Improved Quality, Stability, and Variation [Karras et al. 2017]

GANs Loss Formulations

$$\max_D \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))] + \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log(D(\mathbf{x}))]$$

Discriminator

$$\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

Generator

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_D^{\text{GAN}} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{GAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_D^{\text{NSGAN}} = -\mathbb{E}_{x \sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{NSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_D^{\text{WGAN}} = -\mathbb{E}_{x \sim p_d}[D(x)] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$	$\mathcal{L}_G^{\text{WGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_D^{\text{WGANGP}} = \mathcal{L}_D^{\text{WGAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_g}[(\nabla D(\alpha x + (1 - \alpha)\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{\text{WGANGP}} = -\mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_D^{\text{LSGAN}} = -\mathbb{E}_{x \sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_G^{\text{LSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g}[(D(\hat{x} - 1))^2]$
DRAGAN	$\mathcal{L}_D^{\text{DRAGAN}} = \mathcal{L}_D^{\text{GAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)}[(\nabla D(\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{\text{DRAGAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_D^{\text{BEGAN}} = \mathbb{E}_{x \sim p_d}[x - \text{AE}(x) _1] - k_t \mathbb{E}_{\hat{x} \sim p_g}[\hat{x} - \text{AE}(\hat{x}) _1]$	$\mathcal{L}_G^{\text{BEGAN}} = \mathbb{E}_{\hat{x} \sim p_g}[\hat{x} - \text{AE}(\hat{x}) _1]$

Another problem: Mode collapsing

Other uses

- StackGAN: text to photo

(a) StackGAN
Stage-I
64x64
images

This bird is white with some black on its head and wings, and has a long orange beak

This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face

This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments

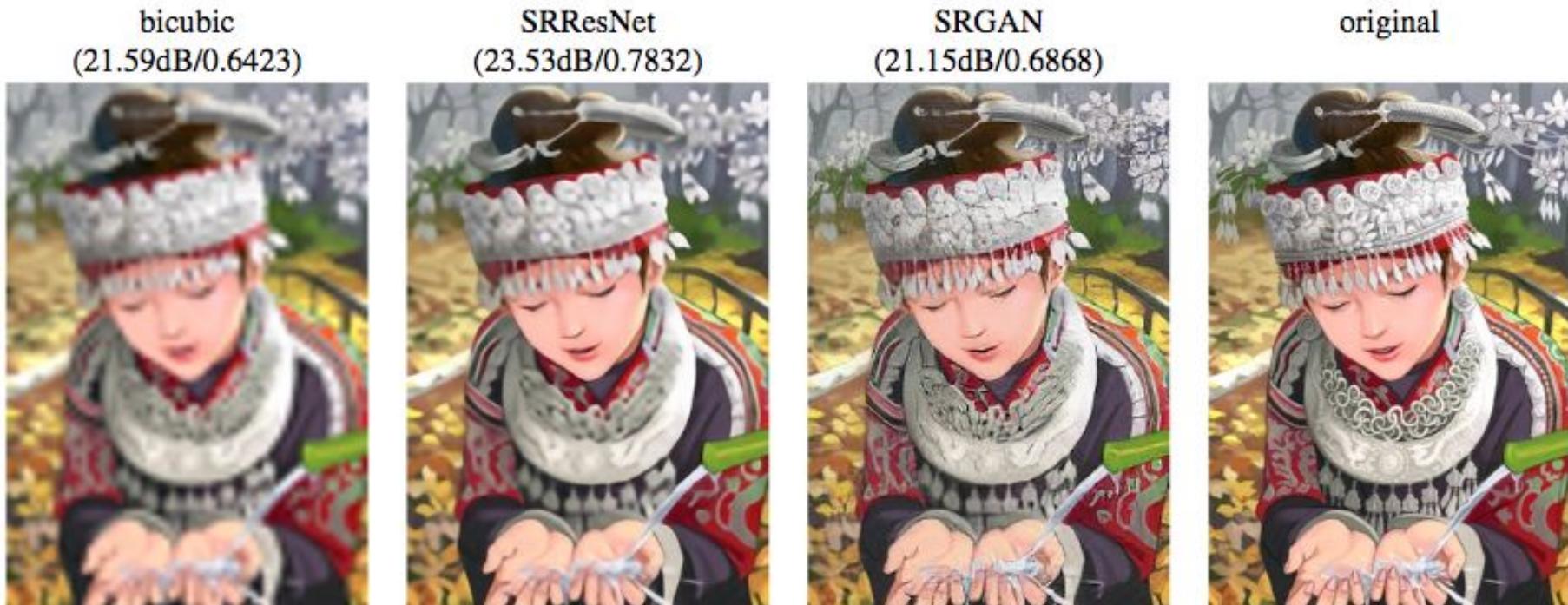


(b) StackGAN
Stage-II
256x256
images

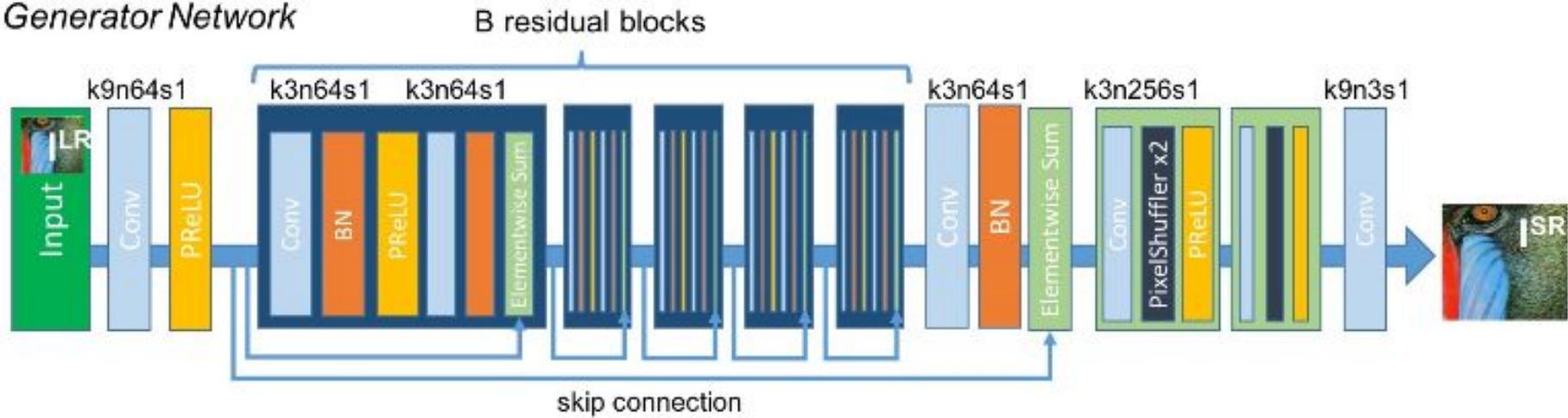
(c) Vanilla GAN
256x256
images

SRGAN

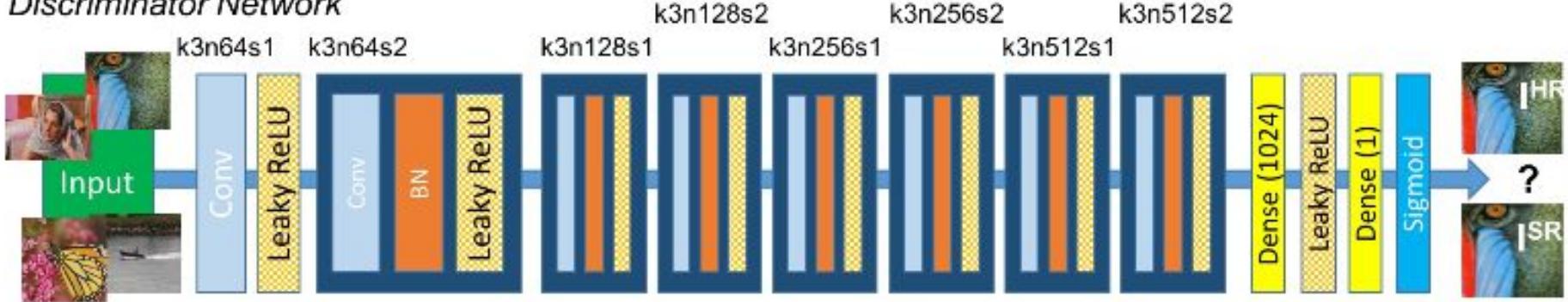
- <https://arxiv.org/pdf/1609.04802.pdf>



Generator Network



Discriminator Network

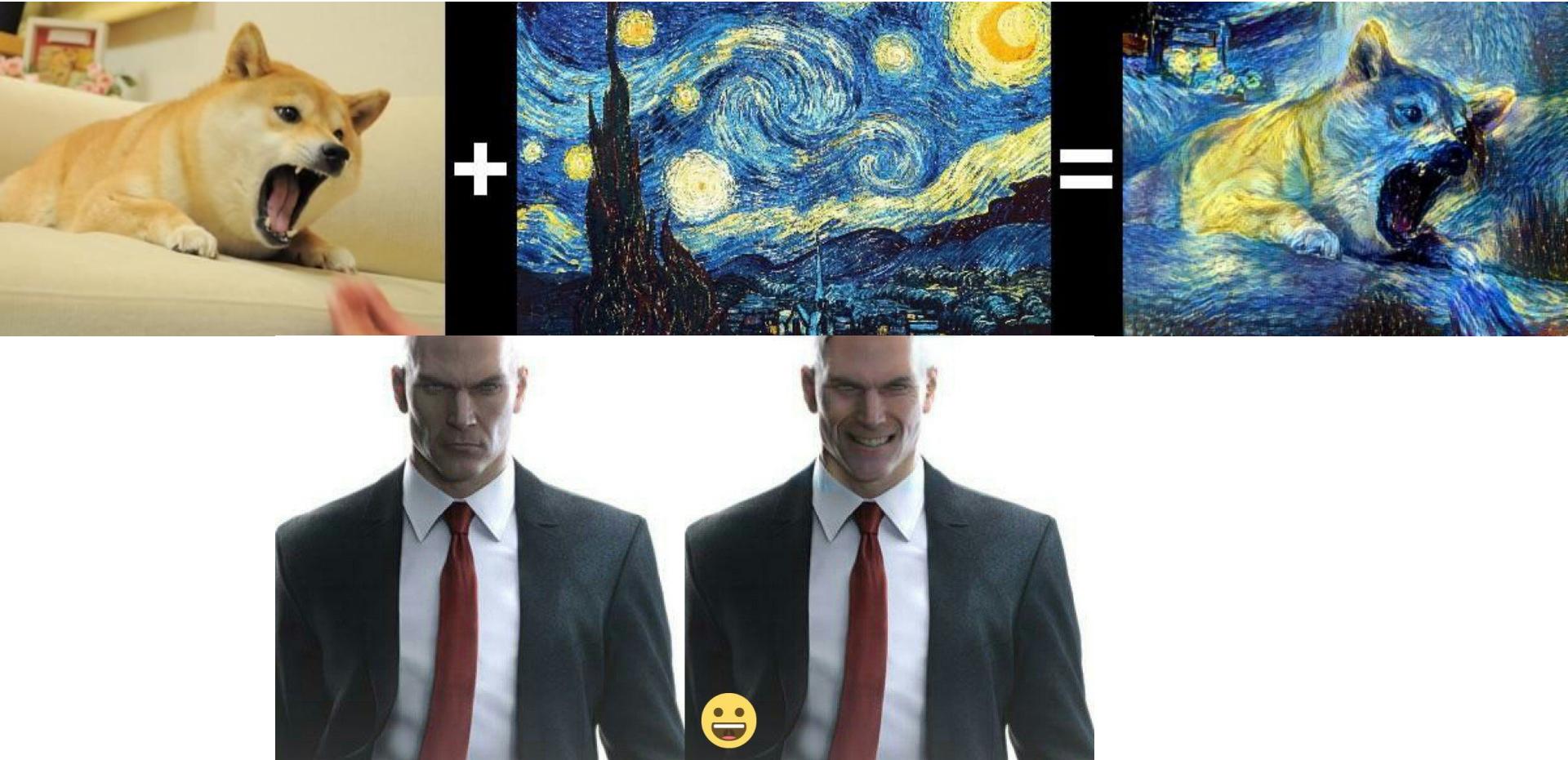


Content-aware Filling



Globally and Locally Consistent Image Completion [Iizuka et al., 2017]

Style transfer with cycleGAN



Zebras  Horses



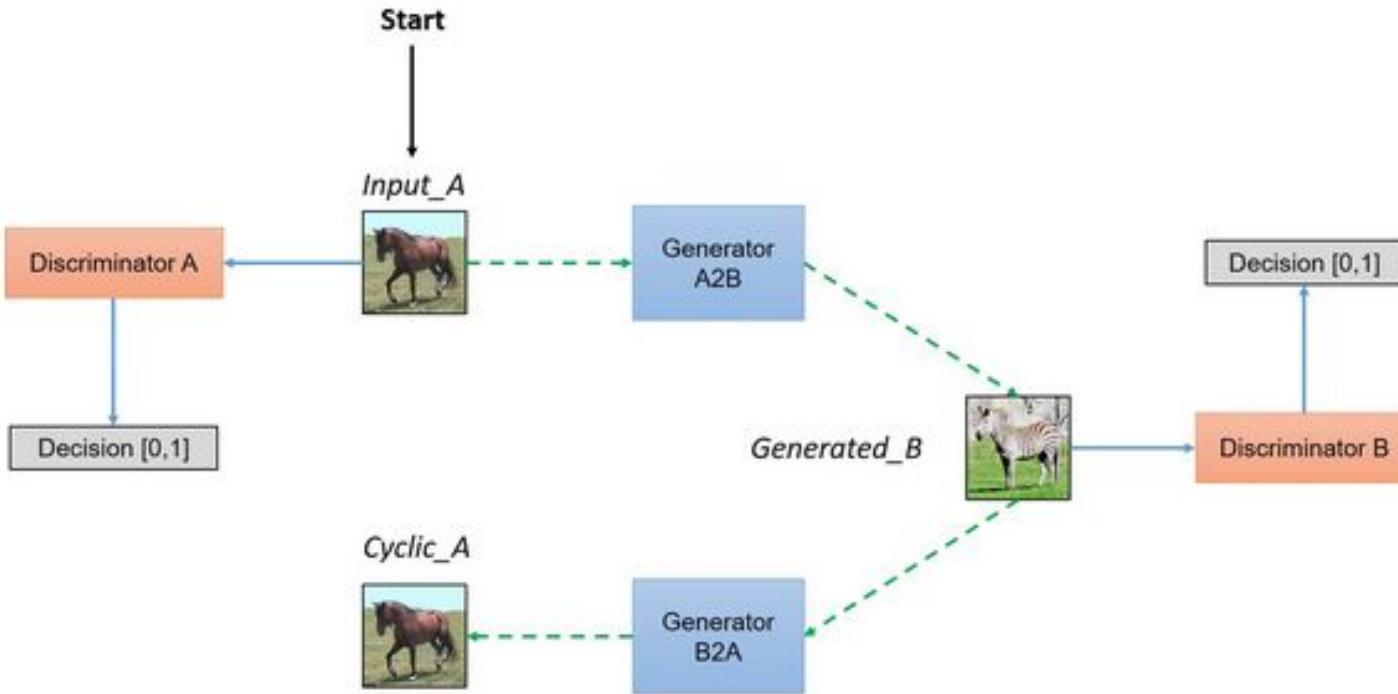
zebra → horse



horse → zebra

Cycle consistency

Note, you don't
need a paired
dataset



Cycle consistency

Machine translation with no parallel text

- MT usually requires parallel text

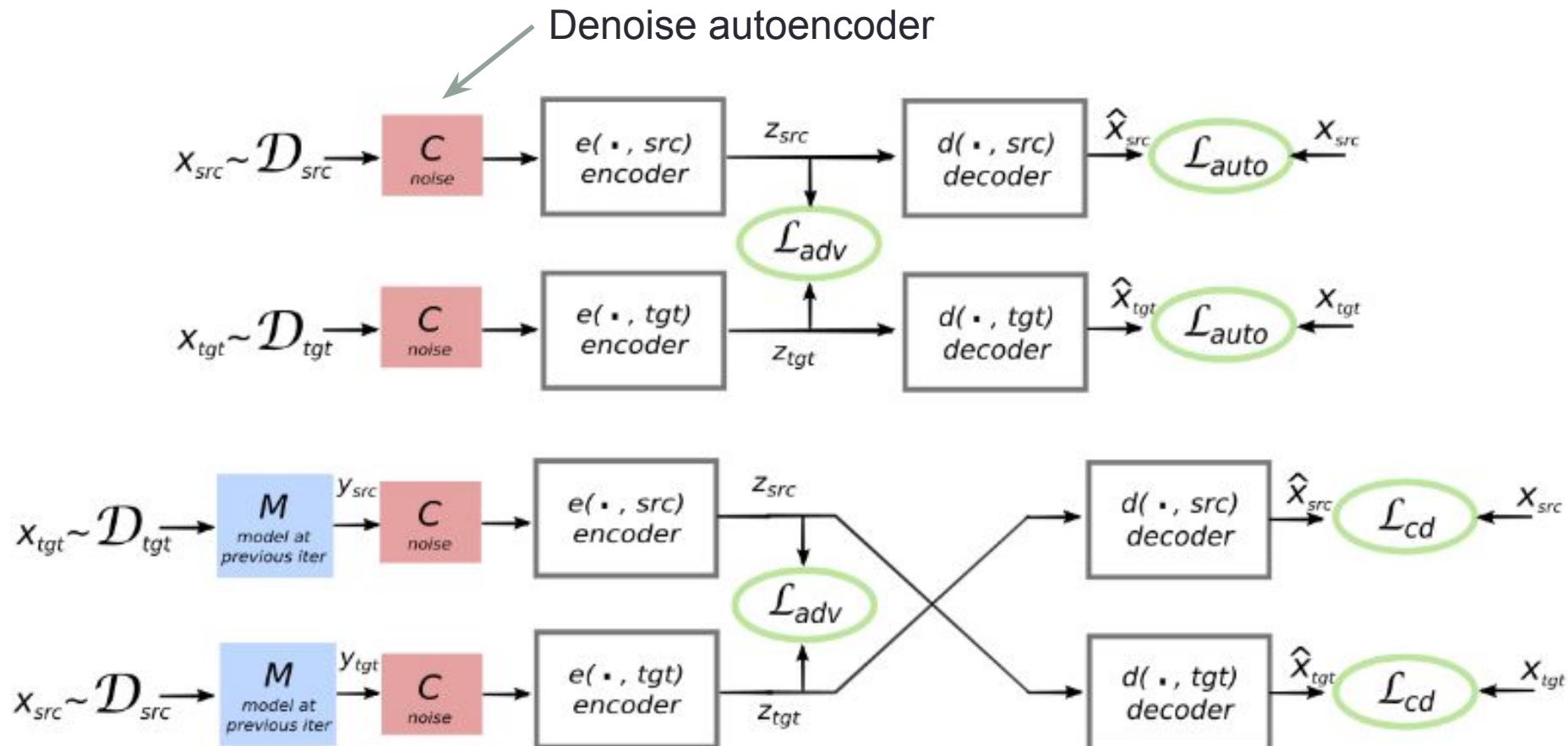
This is a cat

นี่คือแมว

- Most of the time we don't have parallel text. Just text.
- Can we still do MT?
 - Use GANs + Autoencoders

Use GAN Loss (L_{adv}) to enforce that source and target language pairs share the same distributions.

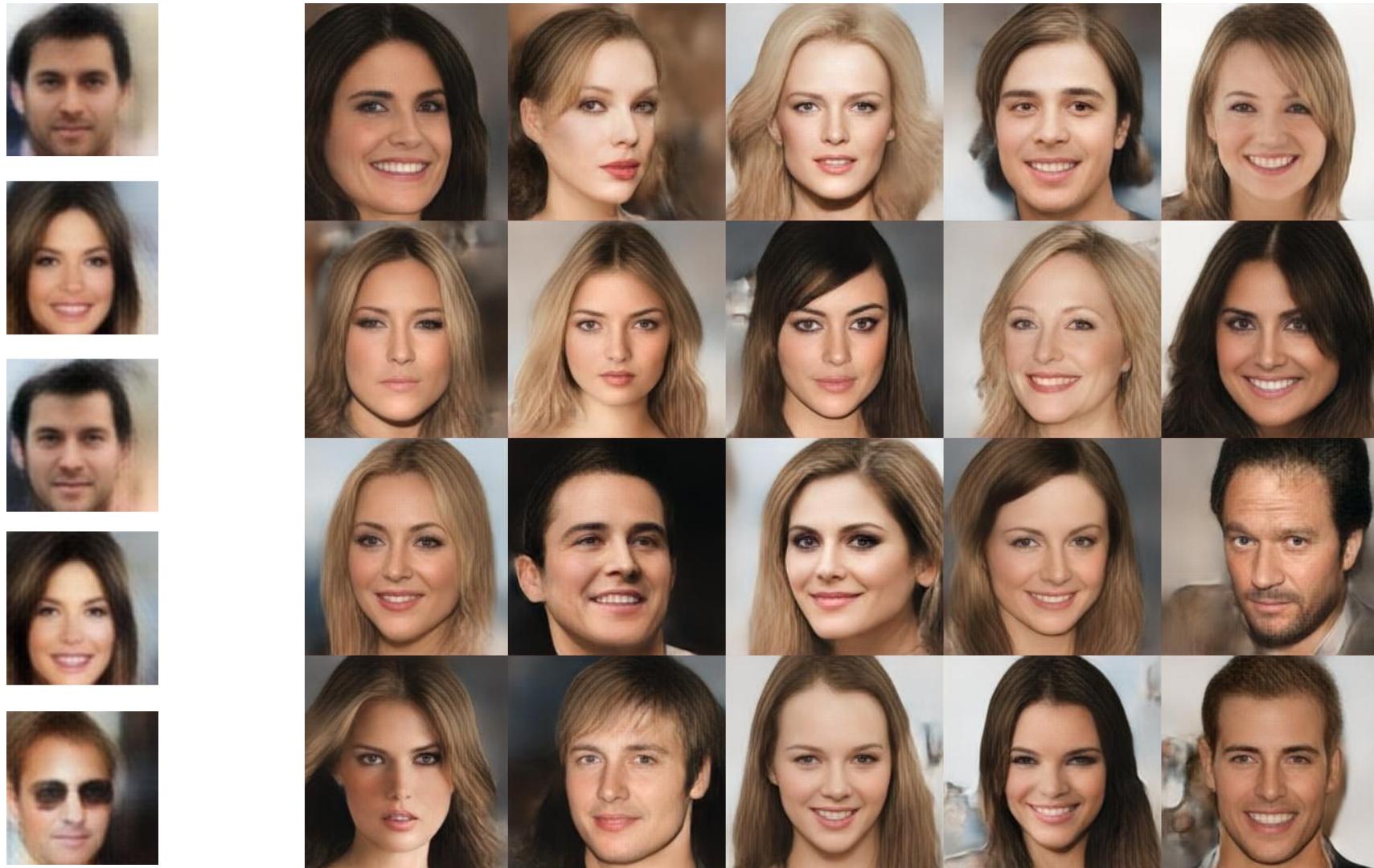
Then enforce the translation distribution matches.



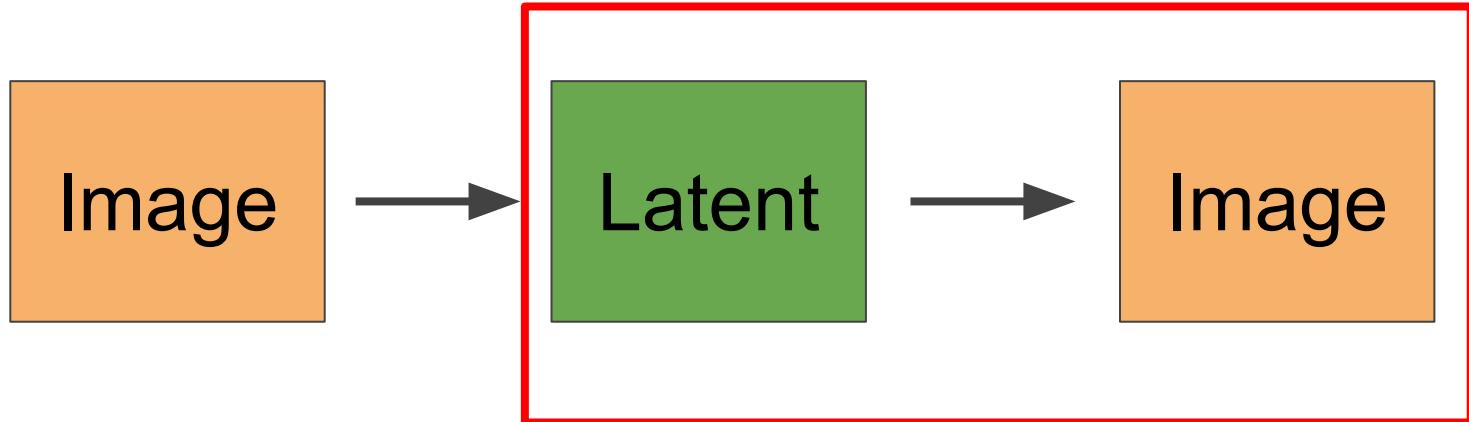
Results

	Multi30k-Task1				WMT			
	en-fr	fr-en	de-en	en-de	en-fr	fr-en	de-en	en-de
Supervised	56.83	50.77	38.38	35.16	27.97	26.13	25.61	21.33
word-by-word	8.54	16.77	15.72	5.39	6.28	10.09	10.77	7.06
word reordering	-	-	-	-	6.68	11.69	10.84	6.70
oracle word reordering	11.62	24.88	18.27	6.79	10.12	20.64	19.42	11.57
Our model: 1st iteration	27.48	28.07	23.69	19.32	12.10	11.79	11.10	8.86
Our model: 2nd iteration	31.72	30.49	24.73	21.16	14.42	13.49	13.25	9.75
Our model: 3rd iteration	32.76	32.07	26.26	22.74	15.05	14.31	13.33	9.64

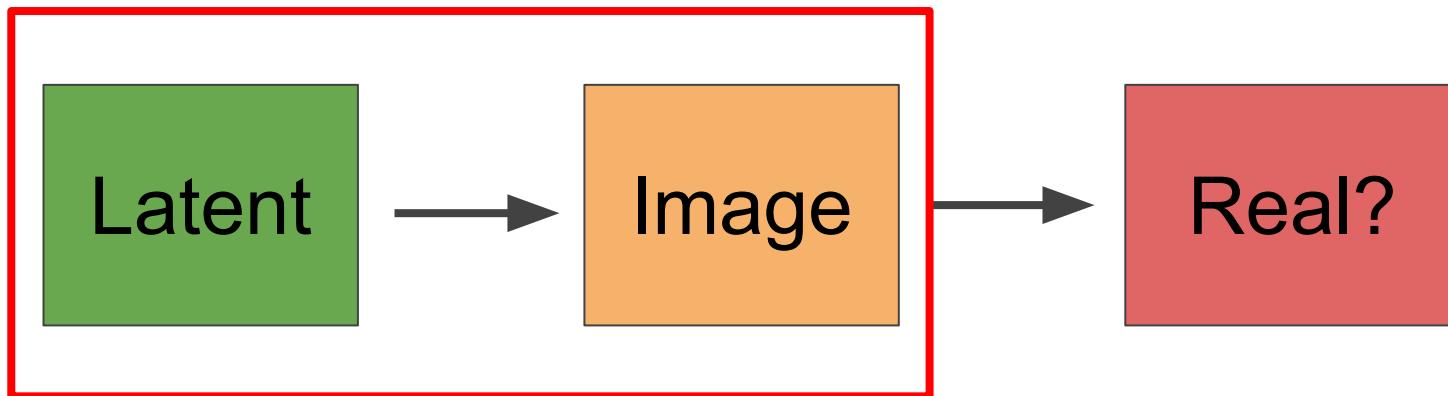
Autoencoder Frontier



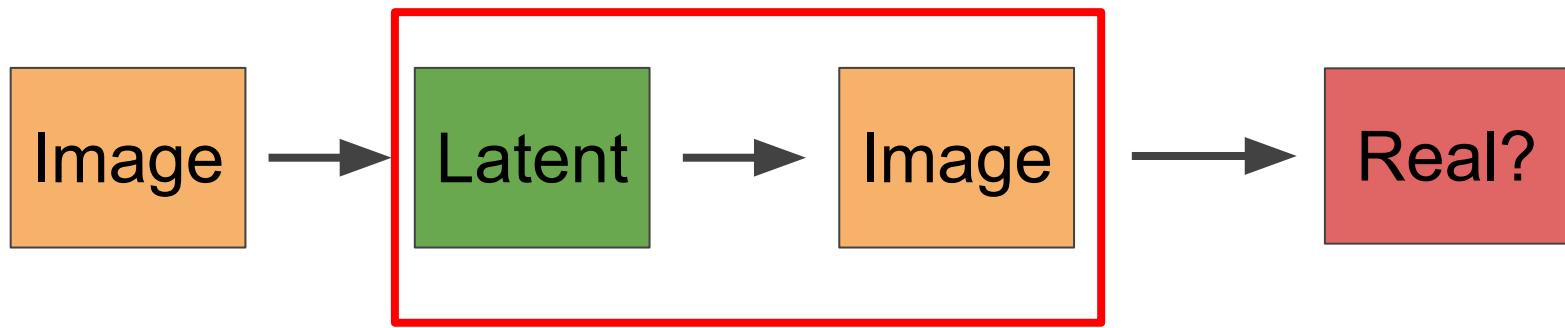
Auto-Encoder



GAN

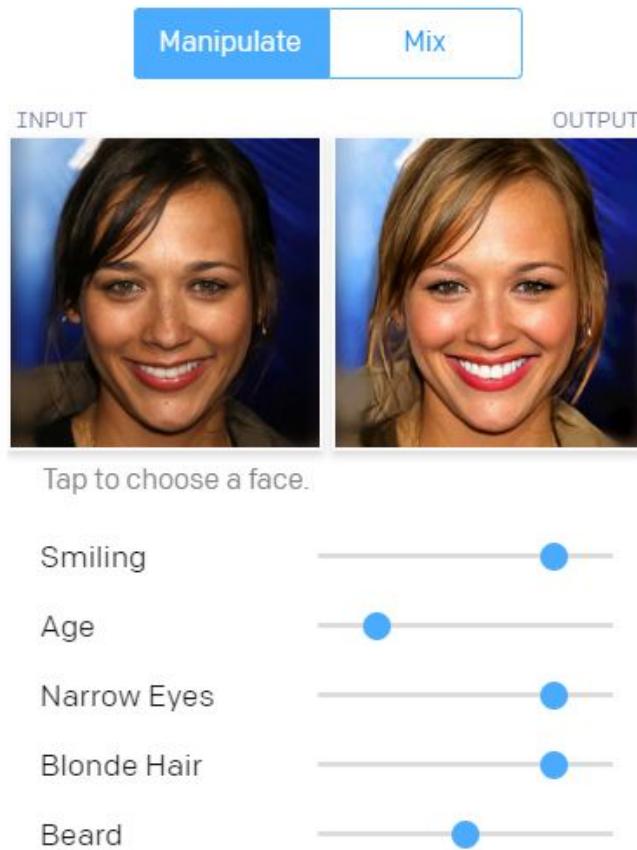


Auto-Encoder + GAN



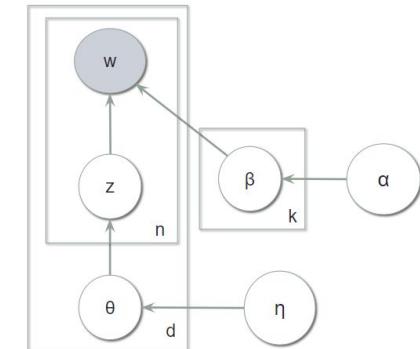
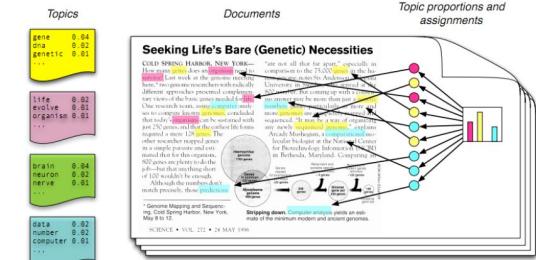
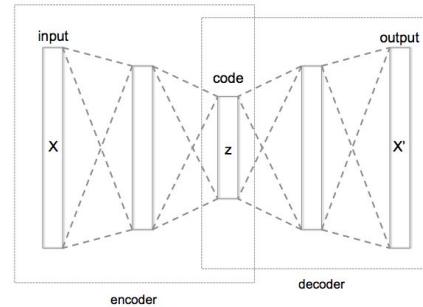
Glow

<https://blog.openai.com/glow/>



Unsupervised Learning Summary

- Clustering
 - Introduction to probabilistic graphical models
 - LDA
- Autoencoders
 - Denoising Autoencoder
 - VAE
 - Evidence lower bound
- Generative Adversarial Networks
- Random Cool stuff



Unsupervised Keypoint Discovery

Reminder

Next class: practical issues in machine learning application deployments, e.g. more various random stuff

Next next class: project presentation with TWO industry judges :)